UNIVERSIDAD CARLOS III DE MADRID

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TRABAJO FIN DE GRADO

ALGORITMOS DE LENGUAJE MÁQUINA PARA LA DETECCIÓN DE VEHÍCULOS MEDIANTE CÁMARAS INFRARROJAS

Autora: Carmen Burguillos Sánchez

Tutor: Arturo de la Escalera Hueso

DEPARTAMENTO DE INGENIERIA DE SISTEMAS Y AUTOMATICA

Leganés, Septiembre de 2016

Agradecimientos

En primer lugar agradecer a mi tutor, Arturo de la Escalera, el cual me permitió adentrarme en esta aventura tan fascinante, como es el mundo de la visión por computador y sus aplicaciones en los vehículos autónomos. Gracias a él no hubiera llegado hasta lo que he conseguido, y sobre todo, no hubiese aprendido todos estos nuevos conocimientos.

Me gustaría agradecer a mis amigos y a mi pareja, los cuales han sufrido esta dura etapa conmigo y me han ayudado a rebasar los problemas cuando parecía que no tenían solución. Gracias por ayudarme en todo lo que estaba a vuestro alcance, mostrarme vuestra confianza en mí, por todo vuestro apoyo y por todos esos grandes momentos que me ayudaron a no desistir. Sin duda, los llevaré siempre conmigo.

Por último quiero agradecer a mis padres, a mis hermanos y a mi abuela, los cuales desde el inicio de la carrera han mostrado su incondicional apoyo y me han ayudado a seguir luchando. Habéis sido una pieza muy importante, no sólo durante la realización de este gran proyecto, sino también en el transcurso de mi carrera universitaria.

Gracias a todos lo que confiaron en mí.

Carmen Burguillos Sánchez
Septiembre 2016

Índice

Tabla de figuras

Resumen

Este trabajo de final de grado consiste en la detección de vehículos gracias a algoritmos de lenguaje máquina, los cuales a su vez, gracias a la visión artificial, han podido conseguir, generalmente, la detección de turismos aunque, también son capaces de reconocer autobuses, camiones o furgonetas. El trabajo ha consistido en la generación de varios clasificadores que cumpliesen con la condición de detectar vehículos bajo ciertos criterios, como la distancia de separación entre ellos o el tiempo de cómputo, los cuales serán explicados posteriormente.

Este estudio podría clasificarse dentro de lo que se denomina Sistemas Avanzados de Ayuda a la Conducción (SAAC), los cuales, gracias a las nuevas tecnologías, pueden ser calificados como los nuevos sistemas de seguridad que ayudarán a una mejor conducción, tanto en un vehículo pilotado como no pilotado. Es por ello, que ha sido realizado junto con el departamento de Sistemas y Automática de la Escuela Politécnica de la Universidad Carlos III de Madrid. La idea surge a partir del desarrollo del coche autónomo gracias al Laboratorio de Sistemas Inteligentes [1], de esta misma universidad, la cual tiene entre sus investigaciones este proyecto. Se han realizado numerosas aplicaciones para mejorar el sistema de conducción asistida que posee este vehículo, y este trabajo podría llegar a ser una más, ampliando así el rango de objetos que el coche es capaz de reconocer y detectar.

Gracias a los dispositivos que ya tiene incorporado este, tales como sensores y cámaras, se ha podido desarrollar esta investigación, ya que han sido la base del trabajo realizado. Se ha hecho uso de las secuencias obtenidas por la cámara infrarroja incorporada y cabe mencionar que no ha sido necesario el uso de iluminación externa, puesto que el sistema de infrarrojos es invariable a los cambios de iluminación.

Con el fin de encontrar el mejor detector que se amolde a nuestros requisitos, se han utilizados distintos métodos para los entrenamientos, haciendo uso así de las librerías OpenCV o SVMLight para aplicar el método de Viola Jones [2], Haar Cascade, o el de la combinación de Histogramas de Gradientes Orientados junto con las máquinas de vector soporte, según queda recogido en la investigación de Dalas y Triggs [3].

1 Introducción

1.1 Motivación

A pesar de los grandes cambios que se han conseguido en el área de la automoción, todavía existen muchos problemas a tratar, como puede ser el caso de la movilidad, la eficiencia del consumo o la seguridad en la conducción. Según un estudio de la OMS, Organización Mundial de la Salud [4], se producen alrededor de 1.25 millones de muertes al año en el mundo por accidentes de tráfico, lo cual, a pesar de ser una cifra mucho inferior a previos estudios, sigue siendo algo que llama al ser humano a estudiar, investigar y corregir.

La conducción no siempre es segura, las distracciones a las que el ser humano está expuesto, los distintos factores a tener en cuenta, las condiciones ambientales del momento... Todo esto hace que se dificulte esta actividad que es tan cotidiana, y necesaria a día de hoy en el mundo. Con la finalidad de eliminar estos problemas, se creó una comunidad especializada en este tipo de investigaciones, la comunidad de Sistemas Inteligentes de Transporte [5] (de sus siglas en inglés, ITS).

El concepto de ITS está centrado en la aplicación de las nuevas tecnologías, tales como la informática y las telecomunicaciones, al mundo de la conducción con el fin de mejorar la seguridad en el transporte terrestre. Entre sus numerosos servicios, podemos encontrarnos con, por ejemplo, el cobro en sistemas de peaje, el control de las infracciones en carretera gracias a los sistemas de vigilancia automáticos, o los sistemas de mejora a bordo de los vehículos. Estos últimos están relacionados con la extracción de información del entorno con el fin de facilitar o ayudar a la conducción, es lo que trae el concepto de conducción inteligente.

Esta calificación engloba lo que a día de hoy se le conoce como Sistemas de Asistencia Avanzada al Conductor (de sus siglas en inglés, ADAS) que mejoran la conducción evitando accidentes o minimizando, en la medida de lo posible, el daño sobre los pasajeros que puedan encontrarse a bordo, así como los peatones en las inmediaciones del vehículo, gracias a la obtención de información del entorno a través de los distintos dispositivos con los que el vehículo cuente. Esta idea general es la que además, facilitó el desarrollo del vehículo autónomo, que no es más que una extrapolación de los sistemas recién comentados, que permitan, no sólo, la mejora de la conducción, sino también el control total de este.

Además de la reducción de los accidentes, estos sistemas permitirán otras aplicaciones como podría ser la ayuda a personas con discapacidad. Esto permitiría que pudiesen desarrollar su vida con más normalidad, consiguiendo incluso que los invidentes puedan llegar a conducir de forma autónoma.

El desarrollo de este proyecto es una pequeña ampliación a esa conducción inteligente que se acaba de mencionar. Muchos de los accidentes que se producen son ocasionados por la colisión entre vehículos, y esto, en la mayoría de las ocasiones, es debido a distracciones u obstáculos en la visión. Sin embargo, hay que tener en cuenta las condiciones atmosféricas o de iluminación, ya que no podemos contar con las mismas capacidades en cada momento. Es por ello, que se producen un mayor número de accidentes tras el ocaso, debido a las condiciones de iluminación insuficiente.

De esta forma nació la idea de crear una aplicación que permitiese la detección de vehículos existentes en la vía, para de esta forma alertar al conductor de su presencia, y evitar así los futuros problemas. Esto se realizaría con cámaras infrarrojas, y así cubrir aquellas horas del día en las que la iluminación en el ambiente no es la suficiente para captar correctamente el medio.

1.2 Objetivos

Tras lo explicado anteriormente, es evidente que ante circunstancias como las mencionadas y con el fin de mejorar la calidad de vida del ser humano, el uso de la visión artificial podría considerarse no sólo útil, sino también necesaria. De este modo, lo que se pretende con el proyecto es realizar un estudio destinado a la obtención del mejor método, o al menos el que realiza un trabajo con mejor precisión, para la detección de vehículos mediante una cámara infrarroja. Esto conllevará una serie de objetivos en el trabajo, tales como:

- Obtención de las imágenes necesarias para los entrenamientos así como, para las pruebas del detector.

- Selección y etiquetado de estas imágenes, de forma que obtengamos las capturas exactas de los vehículos, las cuales se utilizarán posteriormente para el entrenamiento.

- Realización de distintas pruebas mediante cambios de parámetros específicos, con el fin de estudiar los distintos métodos.

- Análisis de los resultados obtenidos y conclusiones.

Es por ello que para el cumplimiento y desarrollo de estas ideas, y conociendo los dispositivitos con los que se va a trabajar es necesario tener en cuenta otros aspectos como:

- Las imágenes pueden ser tomadas de día o de noche, aunque es recomendable una vez caída la tarde para evitar las altas temperaturas que en

algunos momentos del día, en la época de verano, pueden llegar a confundir la percepción de la cámara infrarroja.

- Las imágenes pueden ser de cualquier tamaño, aunque en nuestro caso trabajaremos con imágenes bastante pequeñas, 164x132 píxeles, ya que son las proporcionadas por la cámara. En cuanto a la calidad de las imágenes, cabe destacar que cuanto mayor sea la calidad, mejores resultados se obtendrían.

- La aplicación debe ser capaz de detectar vehículos a distintas distancias, aunque aquellos que se encuentren en una posición muy lejana no serán necesarios.

- Las imágenes podrán ser tomadas desde distintos ángulos, y obtener una composición de ambas imágenes. Sin embargo, en nuestro caso, la cámara está situada en una posición fija.

Para realizar todo lo mencionado ha sido necesario seguir una serie de pautas para realizar las tareas correctamente y de forma ordenada:

- Lo primero a realizar fue la documentación sobre el proyecto a desarrollar. Puesto que este se basó en el análisis de distintos métodos, era necesaria la creación de varios caminos o alternativas para así conseguir la solución final, la detección de vehículos. Para ello, se realizó una investigación en detectores y en los posibles métodos a utilizar.

- Acto seguido, fue necesaria la instalación de los distintos programas o librerías necesarias, tales como OpenCV, SVMLight o QtCreator, el cual ha sido el entorno sobre el que hemos trabajado. El lenguaje de programación utilizado, por lo general, ha sido C++, aunque también se han utilizado ciertos Scripts en Python para la edición de las imágenes.

- Una vez editadas las imágenes, se realizaban los entrenamientos y por último se comprobaban esos detectores con imágenes de testeo, es decir, imágenes que no fueron utilizadas para los entrenamientos, con el fin de ver la verdadera aplicación en la vida real con un conjunto de secuencias nuevo.

- Por último, se analizaron y comentaron los resultados.

1.3 Estructura de la memoria

A continuación se explicará la organización de la memoria, así como los contenidos que podrán encontrar en cada capítulo de esta.

- El primer capítulo engloba las partes de objetivos, motivación y estructura de la memoria, lo que nos permitirá hacernos una idea general del proyecto.

- El segundo capítulo corresponderá a la información del Estado del Arte, en el cual haremos una introducción a los vehículos autónomos, englobando desde los sistemas ADAS, hasta su uso en el coche inteligente Laboratorio de Sistemas Inteligentes de la UC3M.

- El tercer capítulo engloba toda la base de fundamentos teóricos de este proyecto. Se explicarán algunos conceptos generales de la visión por computador, como la óptica o los métodos de detección, terminando por una explicación exhaustiva de los métodos de clasificación a utilizar en este proyecto.

- El capítulo 4 recoge todo el desarrollo práctico del proyecto, centrándonos detenidamente en cada método utilizado, y explicando las distintas partes que esto supone, entendiendo estas como entrenamiento, detección y el análisis de los resultados.

- En los siguientes capítulos, quinto y sexto, además de la bibliografía y los anexos, se recogen las conclusiones sacadas del análisis de los métodos utilizados y las futuras investigaciones que se pueden llevar a cabo, el presupuesto de este trabajo, así como los anexos y la bibliografía, como se ha mencionado.

2 Estado Del Arte

Este apartado ofrece la visión de los trabajos previos relacionados con el tema del TFG a tratar, así como el campo de estudio que esto engloba. En las siguientes secciones se pasará a descubrir el origen de los sistemas de detecciones, junto con sus aplicaciones a día de hoy o los sistemas más utilizados. Por último, haremos una breve introducción de los sistemas de detecciones de vehículos existentes a día de hoy y que técnicas se han usado para conseguirse.

2.1 Vehículos Autónomos

Los vehículos autónomos, o robóticos, son aquellos con la habilidad de simular ciertas capacidades humanas de manejo y control del coche. Esto es gracias a los distintos dispositivos y sensores, que tienen incorporados, que les permiten entender el entorno que les rodea y realizar tareas según como ese ambiente sea distribuido. Estos vehículos, aún en desarrollo, se piensa que tendrán la capacidad de reducir bruscamente los accidentes de tráfico gracias a la toma de decisiones de forma inteligente, sistemática y automática. El ser humano tiene sus limitaciones, y este sistema podría eliminarlas. En la figura 2.1 podemos observar el ejemplo del coche autónomo de Google.

Figura 2.1: Coche Autónomo [27]

Para este tipo de vehículos, la detección es la parte más importante de los sistemas incorporados, ya que es la decisión final que permite al automóvil realizar una tarea u otra. Estas detecciones están basadas en elementos del entorno que, con frecuencia, son motivo claro de accidente, como por ejemplo podría ser la detección de peatones, otros vehículos o de señales de tráfico. Para el correcto desarrollo de este tipo de aplicaciones se deben tener en cuenta varios factores como por ejemplo qué tipo de dispositivos utilizar, cómo colocarlos o cómo obtener la mayor información posible, o mediante qué método se va a obtener esa detección de los objetos.

Los sistemas de detecciones de vehículos están pensados para incorporarlos a los

automóviles de forma que se pueda detectar, a tiempo real, cuáles son los obstáculos que nos estamos encontrando en la carretera. Es por ello que el vehículo debe ir bien equipado, con un ordenador lo suficientemente potente para realizar el procesamiento de las imágenes y la detección, así como con los dispositivos adecuados para el correcto funcionamiento.

Entre estos dispositivos cabe mencionar el uso de las cámaras, las cuales juegan un grandísimo papel en la visión artificial, debido a que las imágenes son la mayor fuente de información de la visión por computador. Por ello, es de gran importancia la resolución, la calidad o el tamaño de imagen, así como el modo de enfoque utilizado al tomar las secuencias. Hay que tener en cuenta que, normalmente, a mayor calidad mayor tiempo de procesamiento, por lo que es conveniente, en la mayoría de las ocasiones, encontrar el equilibrio en la relación calidad/tiempo para obtener los mejores resultados posibles en tiempos de reacción deseados.

Sin embargo, a pesar de elegir adecuadamente los mejores dispositivos para nuestra aplicación, en muchas ocasiones, se pueden encontrar distintos problemas como pueden ser la perdida de información de la escena real con respecto a la capturada por la cámara, lo cual puede darnos una imagen heterogénea o distorsionada; movimientos en la cámara, en su mayoría ocasionados por las vibraciones al conducir a altas velocidades; o dificultades en el procesamiento en tiempo real.

La variedad de vehículos, a pesar de cumplir con unas características similares en cuanto a forma, tamaño o los distintos fondos sobre los que se realicen las imágenes, pueden ocasionar confusiones en las detecciones, dando lugar a falsos positivos y negativos. Es por ello, que además de la correcta elección de los dispositivos, es conveniente estudiar cuáles son las necesidades de nuestra aplicación y cómo pueden ser cubiertas mediante los distintos métodos de detección.

2.1.1 Seguridad en los automóviles

Como se ha comentado, las cifras de accidentes de automóviles han sido una llamada de atención al ser humano para moderarlas, por lo que las investigaciones y estudios en torno a la seguridad en los automóviles han aumentado notablemente en los últimos años. Muchos fabricantes de coches han integrado en sus diseños distintos sistemas de seguridad, no sólo para la seguridad de los pasajeros, sino también para la de circunstancias externas, como pueden ser peatones u otros vehículos. De esta forma, se pueden distinguir dos tipos de seguridad asociada a los vehículos:

- Seguridad pasiva: que es la que trata de reducir los daños que pueden sufrir, tanto los integrantes del vehículo, como personas en sus inmediaciones. Este tipo de sistemas no evitan los accidentes, pero ayudan a reducir las

consecuencias, en la medida de lo posible, de estos. Entre ellos podemos encontrar el airbag, cuya función es detener los cuerpos en movimiento, a causa de un accidente, con la mayor suavidad posible, y evitar golpes con los elementos interiores del vehículo. Se estima que son capaces de reducir el riesgo de muerte en un 30%. Junto con este, nos podemos encontrar otros elementos pasivos como pueden ser el cinturón de seguridad, que evita que los integrantes del vehículo salgan despedidos ante una colisión, o el chasis, el cual está estrictamente diseñado para reducir los daños que puedan ser ocasionados a un peatón ante arrollamiento, así como a los pasajeros durante un choque frontal, evitando que la deformación del vehículo llegue al habitáculo donde se encuentran los pasajeros. Como se pueden observar en las figuras 2.2a y 2.2b, estos sistemas reducen, en la medida de lo posible, el daño que sufre el usuario.

(a) (b)

Figura 2.2: (a) Airbags y cinturones [28] (b) Choque frontal [6]

- Seguridad activa: esta, a diferencia de la anterior, está diseñada para evitar los accidentes. Entre estos sistemas nos podemos encontrar el sistema de frenado, como es el caso del ABS, sistema antibloqueo de frenos, el cual, gracias a su diseño, evita que en un frenado brusco, se bloquee la dirección del vehículo, así como, reduce la distancia necesaria para el frenado; el sistema de suspensión, que absorbe las irregularidades de la carretera y evita así que el coche no se comporte adecuadamente; o el control de estabilidad (de sus siglas en inglés, EPS), el cual evita que, cuando el conductor pierda el control, el coche se desvíe del carril y termine saliéndose de la carretera. Su funcionamiento queda mostrado en la imagen 2.3. Gracias a este último, se reduce en un casi 80% las situaciones de peligro, lo que hace a los nuevos vehículos mucho más seguros.

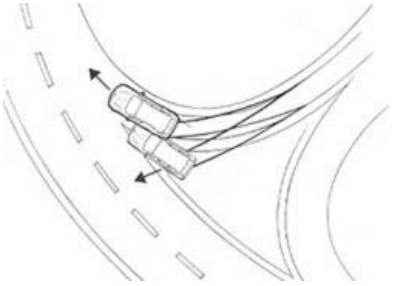


Figura 2.3: Control de estabilidad [6]

2.1.2 Sistemas ADAS

Tras haber definido los tipos de seguridad que engloban a los vehículos, en este caso nos hemos centrado en los coches, pasamos a un nuevo tipo de seguridad que mezcla tanto aspectos de la activa como de la pasiva.

Los sistemas ADAS, o Sistemas Avanzados de Asistencia a la Conducción, son mecanismos que nos permiten mejorar la conducción y reducir así el riesgo de accidente vial, como pueden ser las colisiones con otros. Buscan alertar al conductor de condiciones peligrosas en el camino, el cual podría considerarse una característica de la seguridad pasiva, así como reducir la velocidad o detener el vehículo en caso necesario, tarea que se clasificaría como parte de la seguridad activa.

De esta forma, por ejemplo, nos podemos encontrar con los siguientes sistemas, que actualmente, son utilizados por algunas marcas comerciales de automóviles:

- Sistema de Mantenimiento de Carril: o también conocido como sistema LKAS (Lane Keeping Assit), el cual, gracias a una cámara incorporada en el vehículo, puede monitorizar el trayecto y alertar al conductor en caso de salida del carril. Puede verse su representación en la figura 2.4. Dependiendo del fabricante, algunos coches también llevan incorporada la opción de cambiar levemente la dirección del vehículo automáticamente, cuando esto ocurre, para volver al carril correspondiente, como es el caso de Honda.

Figura 2.4: Sistema LKA [7]

- Control de Crucero Adaptativa: es un sistema que nos ayuda a mantener la distancia de seguridad con los vehículos que se encuentran delante, gracias a un radar de onda milimétrica, como se percibe en la figura 2.5. Tras recibir la información de este, se procesa y se calcula la separación perfecta para evitar colisiones. En el momento que esa distancia no se cumpla, el sistema podría llegar a tener el control para frenar el vehículo, reduciendo así la velocidad de este.

Figura 2.5: Control de Crucero Adaptativa [8]

- Sistemas de Detección: de peatones, ciclistas, motoristas o vehículos, como se muestra en la figura 2.6a. Este sistema, aunque también mejora la seguridad de los integrantes del vehículo, no solo está diseñado para ello. En el caso de los peatones, en la figura 2.6b, ante una situación de atropello, los daños suelen ser mucho peores en estos que en el propio vehículo o en sus integrantes. De esta forma, se pretende reducir esos daños a terceros ante colisión. Estos sistemas trabajan con una cámara, o varias en el caso de tener un sistema de visión estéreo como se comentará posteriormente, que va tomando secuencias en todo momento. Estas son procesadas, y gracias a un código de detección, se puede alertar al conductor ante la presencia de cualquier objeto en el camino del vehículo.

(a) (b)

Figura 2.6: (a) Detección de vehículos [9] (b) Detección de peatones [9]

- Sistema de Reconocimiento de Señales de Tráfico: que consiste en además de en la detección, en la identificación, para de esta manera poder comunicar al conductor de sus significados y, que este, pueda reaccionar rápidamente. Al igual que los sistemas de detección, consiste en una cámara que capta la información a tiempo real, y gracias a una base de datos incorporada en el vehículo, es capaz de, normalmente mediante coincidencia de patrones, identificar de qué señal se trata, como es el caso de la señal identificada en la figura 2.7.

Figura 2.7: Detección de señales de trafico [9]

- Sistema de Asistencia al Aparcamiento: el cual, gracias a unos sensores de ultrasonidos colocados en la trasera del coche, permite alertar al conductor cuando este se acerca demasiado a otro vehículo. Suelen tener incorporado un sistema visual y sonoro para la alerta. Algunas marcas, incorporan además una cámara trasera, como es el caso de Toyota, el cual puede observarse en las imágenes 2.8a y 2.8b, y mediante la cual se puede observar los movimientos que el coche está realizando a tiempo real, evitando de igual manera, las colisiones.

(a) (b)

Figura 2.8: (a) Asistencia al aparcamiento [10] (b) Visión trasera incorporada [10]

- Sistema de Prevención de Colisiones: con un funcionamiento muy similar al de control de crucero adaptativo, este sistema estudia la probabilidad de colisión con el vehículo inmediatamente posterior gracias a la incorporación de radares de ondas milimétricas. En caso de posibilidad de colisión, se alerta al conductor con señales sonoras o visuales, según cada fabricante. Un ejemplo de este sistema es representado en la figura 2.9.

Figura 2.9 Prevención de Colisiones [11]

Además de estos ADAS ya mencionados, también podemos distinguir otros sistemas que están destinados a ofrecer información al vehículo para ayudar a la conducción. Estos son conocidos como IVIS (de sus siglas en ingles In-Vehicle Information Systems), y son, por ejemplo, los sistemas de navegación o de información de tráfico. Estos, gracias a las nuevas tecnologías y a la conexión directa con satélites, como los dispositivos GPS, permiten que los conductores puedan tener un trayecto más tranquilo y evitan, por ejemplo, las distracciones ocasionadas por desconocimiento de la ruta a seguir o los tramos donde existen retenciones o accidentes de tráfico. De esta forma, se consigue de nuevo una conducción más segura.

2.1.3 Laboratorio de Sistemas Inteligentes - Vehículo Inteligente IVVI 2.0

Como se ha mencionado anteriormente el Laboratorio de Sistemas Inteligentes [1] de la Universidad Carlos III de Madrid cuenta, entre otros proyectos, con el vehículo inteligente IvvI 2.0, cuyas siglas en inglés corresponden a Vehículo Inteligente basado en Información Visual, y el cual puede observarse en la figura 2.10

Figura 2.10: IvvI 2.0 [1]

Este vehículo se basa en la obtención de información del entorno que le rodea a través de imágenes, en su gran mayoría, las cuales además son obtenidas por medio de los distintos dispositivos que el coche presenta. Así, a día de hoy, cuenta con los sistemas y sensores más avanzados, así como totalmente integrados, para el desarrollo del Sistema de Asistencia Avanzado al Conductor.

De esta forma, el coche presenta un sistema de visión estéreo, cuya función principal es la detección de objetos, como por ejemplo, es el caso de las señales de tráfico. Además de la cámara infrarroja, que se ha comentado con anterioridad, el vehículo también tiene incorporada una cámara situada en el salpicadero del coche, y dirigida hacia al conductor, la cual permite ejecutar un sistema de prevención de la somnolencia, así como para la identificación de caras. Además del sistema de detección de señales de tráfico, de somnolencia o detección de caras, el vehículo además cuenta con un detector de peatones mediante cámaras infrarrojas, específico para trabajar en un entorno nocturno o la capacidad de diferenciar los carriles de la carretera, pudiendo así alertar al

conductor cuando este se sale de ellos. Otra de las incorporaciones con las que cuenta este coche, es un dispositivo GNSS que proporciona la posición del coche en cada momento, así como el movimiento de este, como se muestra en las figuras 2.11a y 2.11b.

(a) (b)

Figura 2.11: (a) Detección de peatones [1] (b) Localización GPS [1]

Todo esto esta comunicado mediante un sistema CAN Bus, el cual es un bus de comunicación de bajo coste computacional, y es dirigido a un potente ordenador que permite trabajar en tiempo real con toda la información que obtiene del entorno. La arquitectura de software utilizada es la basada en ROS (sus siglas en ingles Sistema Operativo Robótico).

Este prototipo de vehículo inteligente, todavía en desarrollo, es una de las grandes investigaciones de este laboratorio y gracias a los numerosos proyectos que existen en torno a él, y al avance de la visión artificial, en muy poco tiempo podrá verse como ese prototipo de vehículo podrá comercializarse y ayudar a muchos ciudadanos a una mejor conducción.

2.1.4 Legislación

En cuanto a la normativa o la legislación de los coches autónomos, puesto que todavía se encuentra en ámbito de expansión y desarrollo, no cuentan con una legislación concreta. La existente a día de hoy, no recae tanto sobre las tecnologías usadas o sobre límites preestablecidos, sino sobre los factores políticos, jurídicos o de responsabilidad que esto conlleva.

Así por ejemplo en la Unión Europea, a algunas empresas de automóviles, para realizar las pruebas en cuidad, se les conceden ciertos permisos que les otorgan la capacidad de cerrar espacios o controlarlos estrictamente para evitar daños a terceros. De igual forma ocurre en España, ya que en noviembre de 2015, la DGT [12] estableció los límites para la realización de pruebas con los coches autónomos. Como ya se ha mencionado, se necesita un permiso, el cual solo es válido para el territorio nacional, y tiene una duración de 2 años, siendo estos prorrogables.

Gracias a esto, España se convirtió en uno de los primeros países que apoyo el uso de las nuevas tecnologías en los automóviles. Sin embargo cabe mencionar, que a pesar de no existir un control exhaustivo entorno a estos vehículos, cuando lleguen al mercado, las normas viales deberán cambiar para adaptarse a las limitaciones o capacidades que estos coches pudiesen llegar a presentar.

2.1.5 Sistemas actuales de detección de vehículos

Los sistemas de detecciones en carretera, como se ha comentado, basan sus acciones en secuencias de video tomadas a tiempo real en la conducción, con el fin de detectar todo vehículo, o como en nuestro caso, coches en su mayoría, que circule por la misma vía.

Para una mejor extracción de la información, las imágenes son adquiridas y, normalmente, procesadas, para resaltar algunas características de interés frente a otras no tan necesarias, como puede ser el caso de resalto de bordes o aumento de los contrastes, y de esta forma facilitar el trabajo de la detección y permitir obtener mayor precisión en el ejercicio.

La mayoría de los sistemas en funcionamiento a día de hoy se basan en métodos de aprendizaje, como es el caso de este proyecto, pero sin embargo, existen otros caminos como los basados en movimiento o en la visión en estéreo, que también obtienen resultados favorables. Así, se pueden diferenciar tres tipos de clasificadores generales:

- Métodos según el número de fuentes: en el que nos encontramos la visión en estéreo y la monocular, y que se basa en el número de fuentes de información con el que trabaja, y del cual extraen las imágenes.

- Métodos de movimiento: los cuales, a partir del fondo del que se parta, es capaz de diferenciar objetos por los movimientos de estos.

- Métodos basados en la apariencia: entre los cuales destacan los métodos a partir de modelos, y los métodos de aprendizaje, ya mencionados anteriormente.

Cada uno de estos caminos será explicado con más detenimiento en el apartado de visión artificial, puesto que su conocimiento forma parte de los fundamentos teóricos adquiridos para la realización de este trabajo.

3 Fundamentos teóricos

3.1 Visión artificial

Los humanos tenemos la capacidad de poder percibir todo lo que nos rodea y obtener información de ello sin casi realizar ningún tipo de esfuerzo. Está en nuestra naturaleza la capacidad de percibir y comprender todo a lo que nuestros sentidos son capaces de llegar. La visión por computador trata de imitar ese comportamiento peculiar del ser humano, es decir, la visión por computador trata de conseguir que una máquina o computadora pueda acercarse a imitar o a reaccionar como lo harían los humanos.

La visión por computador, o visión artificial, puede definirse como todos aquellos métodos que permiten a un ordenador adquirir, procesar, estudiar y entender lo que se encuentra a su alrededor. Se puede definir como la parte de la inteligencia artificial que basa sus conocimientos en las imágenes captadas del entorno. Es una mezcla de varias disciplinas, como se puede ver en la representación de la figura 3.1, las cuales pueden calificarse en inteligencia artificial, control automático o aprendizaje máquina. Estas son capaces de proporcionar a la maquinas las capacidades de inteligencia computacional, la visión robótica o la visión cognitiva respectivamente.

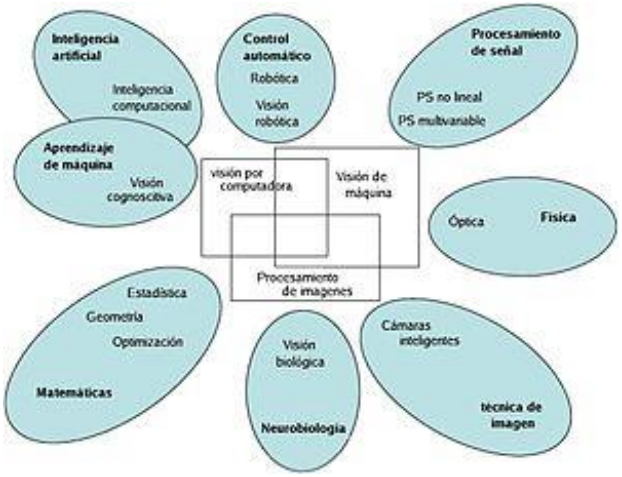


Figura 3.1: Áreas de la visión por computador [13]

Existen numerosas tecnologías que hacen uso de la visión artificial como es el caso de las aplicaciones de detección, reconocimiento de objetos, seguimiento de estos o reconstrucciones de escenas. Así por ejemplo, la visión artificial puede emplearse para el ocio, como al realizar fotos cuando la cámara detecta una sonrisa; para la seguridad, como es el caso de los controles biométricos para el reconocimiento de huellas dactilares, o en temas relacionados con la medicina, como puede ser el reconocimiento de patrones irregulares en algunos estudios que puede prevenir enfermedades como el cáncer. Es un campo muy extenso y del que nos queda mucho por aprender, y que gracias a las nuevas tecnologías está creciendo a grandes pasos.

La comprensión y la percepción que las maquinas adquieren es gracias a las distintas secuencias de imágenes que son obtenidas de cámaras de vídeo. Estas son la base de toda la visión artificial. Los seres humanos captamos la información gracias a la luz que incide en los ojos y es transformada en impulsos nerviosos por la retina. Estos impulsos son trasmitidos al cerebro a través de los nervios ópticos y finalmente, el córtex visual del cerebro permite formar la imagen. Sin embargo las máquinas no pueden realizar esta tarea sin la ayuda de un aparato externo que sea capaz de captar del entorno. Las cámaras, conectadas a estas computadoras, proporcionan el sentido visual y permiten leer el ambiente.

A pesar de haber avanzado mucho, las cámaras no realizan la función exacta que podría desempeñar el ojo humano, y con frecuencia distorsionan la realidad al plasmarlo en una imagen. Esto, en la gran mayoría de las ocasiones, es debido a la aparición de ruido en las imágenes, el cual no es más que alteraciones en el valor de los píxeles a la hora de procesar la información en la realización de la imagen. Aunque existen distintos tipos de ruido, podemos destacar dos, los cuales son los más significativos:

- Ruido *salt and pepper*
  Es el ruido más complicado de eliminar. Consiste en la combinación de píxeles blancos y negros en toda la imagen, siendo ese el motivo, por el cual se le conoce con ese nombre, *salt and pepper*, o su traducción, sal y pimienta. Normalmente está causado por la saturación del dispositivo o la pérdida de conexión de este, lo que ocasiona que la información del píxel correspondiente en ese momento, se vea afectada por los acontecimientos. El valor que toma estos pixeles anormales puede ser 0, tomando el color negro, o 255, siendo en este caso blanco. En la figura 3.2 puede observarse como este fenómeno afecta a una imagen.

Figura 3.2: Ruido *Salt and Pepper* [14]

  Este tipo de ruido puede ser eliminado gracias a filtros no lineales como el de la mediana, que queda representado en la figura 3.3, y el cual, al coger los valores de un píxel y sus vecinos y ordenándolos de menor a mayor, siempre va a dejar

estos píxeles defectuosos fuera del nuevo valor. Se elige el valor que ocupa la posición central de ellos, o en su defecto al ser un número par de píxeles, se toma la media de los dos valores centrales.

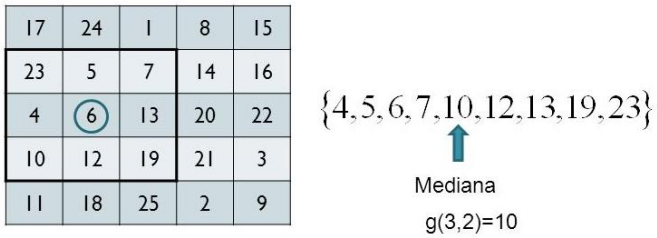


Figura 3.3: Filtro de la mediana [15]

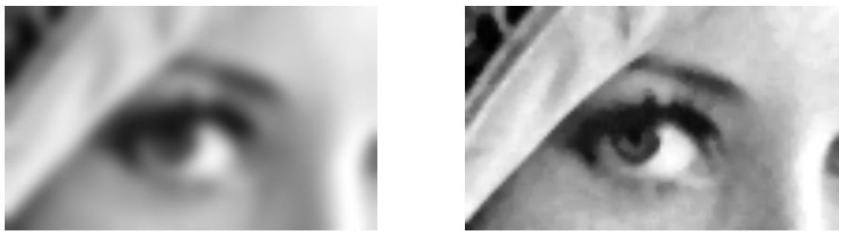
- Ruido gaussiano

  El caso del ruido gaussiano es bastante distinto al anterior. El valor de los píxeles será el valor original más un valor añadido por defecto debido al aparato.

$$\boldsymbol{f'(x) = f(x) + \delta} \tag{3.1}$$

$$siendo:$$
$$f'(x): el\ nuevo\ valor$$
$$f(x): valor\ original$$
$$\delta: valor\ adicional$$

  Los nuevos valores de estos píxeles tenderán a ser más grises en comparación con el salt and pepper. Este tipo de ruido es relativamente fácil de eliminar gracias a los filtros gaussianos, sin embargo estos también ocasionan un suavizado en la imagen, lo que significa que la imagen dejará de estar definida, y que por lo tanto los bordes no serán tan abruptos, como se puede percibir en las figuras 3.4a y 3.4b.



(a) (b)

Figura 3.4: (a) Filtro Gaussiano [16] (b) Filtro de la Mediana [16]

Con el fin de eliminar las imperfecciones del ruido, así como otros defectos, tales como la distorsión de la imagen, la falta de contraste iluminación insuficiente o la necesidad de modificación de alguna característica, antes de trabajar con las imágenes, en cualquier aplicación se hace un procesamiento de esas, lo cual permite realizárselos.

Tras esto, se pasa al análisis de las imágenes, en el caso de querer extraer alguna información concreta de esta. Un ejemplo claro podría ser el caso de un control de calidad en una cadena de montaje. Se suele incorporar una cámara, normalmente junto con iluminación externa, que permita captar los movimientos de las piezas de la cadena o el resultado final de estas. Así, cuando la pieza ha finalizado, se realiza un chequeo informático para poder seguir con el proceso, o de lo contrario desechar a pieza por defectuosa.

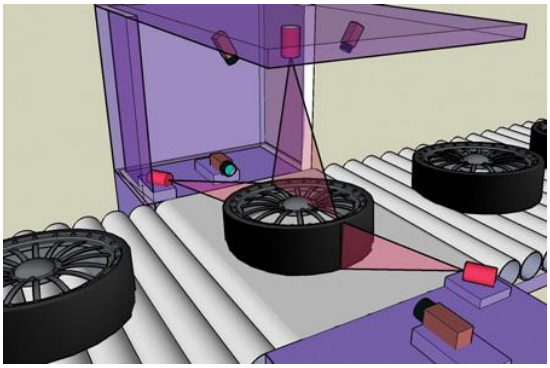


Figura 3.5: Control de calidad automatizado [17]

Por ejemplo, en el caso de la figura 3.5, podemos observar lo que sería el control de calidad de los neumáticos de la imagen. De esta forma, con los dispositivos que presenta, se podrá monitorizar el proceso completo, así como visualizar la calidad del trabajo previo. En este caso, podría ser la comprobación del grosor de la goma o la correcta colocación del tapacubos.

Tras esto, podemos observar que la visión artificial tiene un amplio rango de aplicaciones y, como se ha mencionado, actualmente está en auge el uso de esta. A continuación se pasará a explicar la aplicación que esta tiene en el ámbito de los vehículos autónomos, entrando así en más detalle en este proyecto.

3.1.1 Óptica y cámaras

La óptica tiene un trabajo muy importante en la visión artificial, ya que sin las cámaras esta perdería su capacidad visual, como ya se ha comentado. La óptica de una cámara tiene como misión capturar todos los rayos de luz y centrarlos en un elemento sensitivo de la cámara. El tipo de óptica que se debe utilizar deberá depender de la calidad de imagen que queramos obtener o del tamaño del objeto final.

En todo modelo óptico existen unos parámetros comunes los cuales son:

- Punto focal: es el punto donde los rayos convergen.

- Distancia focal: distancia entre el eje de la lente y el punto focal. Gracias a este

parámetro podemos calcular la posición y el tamaño de los objetos en la imagen.

- Numero F: es un parámetro que relaciona la distancia focal con la apertura del diafragma.

$$\boldsymbol{F = \frac{f}{D}} \tag{3.2}$$

  Si F es pequeño, estamos dejando pasar mucha luz. Por el contrario, a mayor F, menos luz dejaremos pasar y más oscura será la imagen.

De esta forma pasaremos a explicar algunos parámetros básicos a tener en cuenta de la óptica, así como los modelos existentes para la obtención de las imágenes.

3.1.1.1 Modelo de lente fina

Este modelo cuenta con una lente muy fina, la cual atraviesan los rayos de luz. Desde la lente, a un lado se encuentra el plano del objeto y los puntos relevantes para nuestra imagen, y al otro lado se encuentra el plano de enfoque, o el plano del a imagen. El eje óptico es la línea imaginaria que atraviesa perpendicularmente la lente por un punto central.

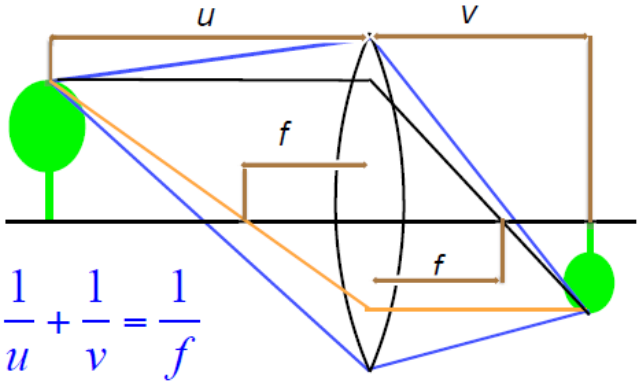


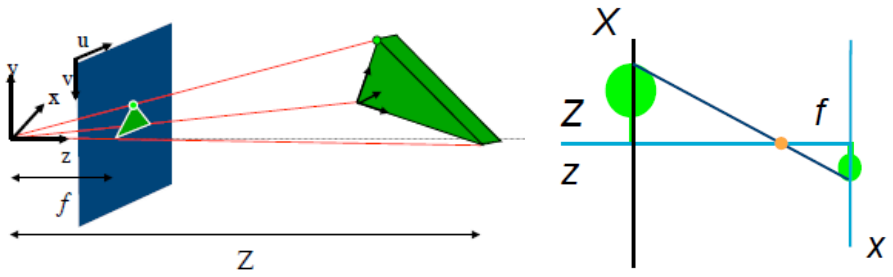
Figura 3.6: Modelo de Lente Fina [18]

Estos entran paralelos al eje óptico horizontal desde los puntos a querer plasmar en un plano, y tras atravesar la lente, convergen en el punto focal. De igual manera desde el lado del plano de enfoque, los rayos van paralelos al eje óptico y al atravesar la lente, convergen en el punto concreto. Este modelo cuenta con una ecuación matemática, gracias a las relaciones de los triángulos semejantes, que puede observarse en la figura 3.6.

3.1.1.2 *Modelo Pinhole*

El modelo de *pinhole* define una relación geométrica entre un punto en tres dimensiones y su correspondiente proyección en dos dimensiones en un plano de la imagen. Esta transformación es lo que se llama proyección perspectiva.

Este es una cámara muy simple, que no posee lente y que solo dispone de una pequeña apertura. Gracias a esta apertura, la cual además le da el nombre al modelo (*pinhole*, y su traducción del inglés "ojo de aguja"), la luz puede pasar. Cuando esto ocurre, se forma una imagen invertida en el lado opuesto, de igual forma que ocurre con el ojo humano. Cuanto más pequeño es el agujero, más nítida se verá la imagen, sin embargo, de igual forma, se verá más oscura. Este tipo de modelo necesita más tiempo de exposición que otros, debido al pequeño agujero. En este caso, al distancia focal es la distancia entre el agujero a la imagen que se forma.



(a) (b)

Figura 3.7: (a) y (b) Modelo de *Pinhole* [18]

Este modelo es muy sencillo y puede asemejarse a una caja con un orificio por el que pueda pasar la luz. Gracias a este diseño, es uno de los modelos más utilizados en las cámaras comerciales. De la relación entre las coordenadas en el espacio y las de su proyección en el plano, como se pueden observar en las figuras 3.7a y 3.7b, se obtienen las siguientes relaciones matemáticas, por trigonometría, en las que se basa este modelo.

$$\mathbf{y} = \frac{\mathbf{f}}{\mathbf{Z}}\mathbf{Y}\ ;\ \mathbf{x} = \frac{\mathbf{f}}{\mathbf{Z}}\mathbf{X} \tag{3.3}$$

3.1.1.3 *Cámaras Infrarrojas*

Como se ha comentado el coche autónomo de la UC3M tiene incorporado una cámara infrarroja con la que se han obtenido todas las muestras necesarias para este proyecto.

Las cámaras de infrarrojo, o cámaras térmicas, son dispositivos que se encargan de transformar el calor en luz para así poder analizar los objetos. La imagen resultante es lo que se conoce como termograma y es analizado a través de un proceso llamado termografía. Un ejemplo de este caso puede verse en la figura 3.8. Estos dispositivos

toman la temperatura del entorno y lo plasman en una imagen, mostrando así la diferencia de temperatura entre las distintas superficies. La función de estas cámaras no es sólo observar, sino también transformar esa información captada en una imagen digital, que después puede ser guardada, procesada o mostrada por un monitor. Es por ello que las cámaras infrarrojas son utilizadas ampliamente en numerosas aplicaciones tales como vigilancia, navegación o visión por computador.

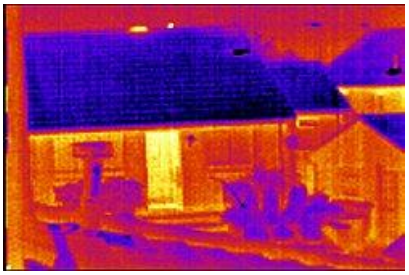


Figura 3.8: Termograma [19]

Estas cámaras son equipos especializados para conseguir apreciar lo que el ojo humano no puede. La única banda que podemos captar es el de la Luz Visible, la cual recoge longitudes de onda que emiten mucha más energía que la banda del infrarrojo. Concretamente la cámara dispuesta en el coche está especializada en el infrarrojo lejano o *Far Infrared*, la cual engloba unas longitudes de onda en el espectro infrarrojo de la radiación electromagnética desde los 5 micrómetros hasta 1mm, como se aprecia en la figura 3.9.

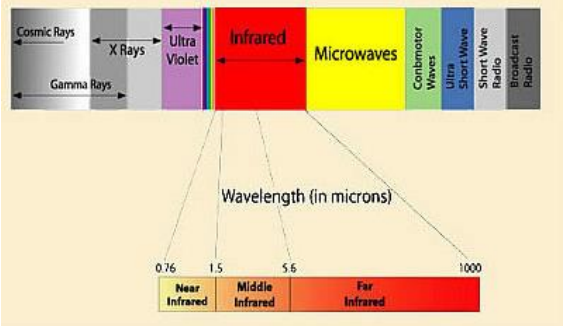


Figura 3.9: Espectro de luz [20]

Cualquier objeto con temperatura mayor a 0ºK emite radiación infrarroja, y esta puede ser recogida por esta tecnología, permitiéndonos así obtener un amplio rango de detalles. Además, estas cámaras captan un gran contraste en oscuridad completa y son capaces de trabajar bajo luz del día, así como niebla, humo o polvo, lo cual es una gran ventaja, y además, fue una de las principales motivaciones para trabajar con ella frente a cámaras convencionales o de visión nocturna.

Trabajar con las cámaras convencionales limitaba bastante los resultados. Estas no son invariantes a los cambios de iluminación, y durante la noche, cuando el conductor puede

perder más la concentración, era necesario una iluminación externa muy potente. Esto es porque las cámaras convencionales, al igual que las nocturnas, captan luz y no temperatura. Los vehículos, al igual que las personas o los animales, tienen una temperatura elevada si los comparamos con el entorno, por lo que es muy sencillo distinguirlos gracias a este tipo de tecnologías. Es por ello, que la idea de las convencionales fue desechada desde un primer momento.

Con respecto a las cámaras de visión nocturna, estas tendrían un comportamiento similar. Aunque estas sí están especializadas para trabajar en entornos oscuros, no obtienen resultados tan precisos como las infrarrojas. Además, en el caso de encontrarnos frente a un entorno de oscuridad total, la cámara de visión nocturna es inservible. Estas necesitan de una iluminación mínima para poder contrastar toda la panorámica. Sin embargo, las cámaras infrarrojas seguirían realizando una magnífica labor ya que no se ven afectadas por el cambio de iluminación. En la figura 3.10, puede verse un ejemplo de este tipo de cámara.

Figura 3.10: Visión Nocturna [21]

El uso de las cámaras térmicas quedó definida por lo recién comentado, pero sin embargo, esto contaba con un inconveniente principal, la temperatura. A pesar de ser el mejor método para obtener información del entorno, bajo ciertas condiciones podía no funcionar correctamente. Así por ejemplo, si la temperatura de ambiente es muy elevada y el fondo sobre el que está el coche tiene una temperatura similar a la del vehículo, en la imagen de salida no existirá diferencia entre ellos. Es por ello, que en estas situaciones será necesario recurrir a otros dispositivos como las cámaras convencionales o los sensores de ultrasonido, para poder detectar si existen un vehículo delante del coche o no.

Otro inconveniente, aunque no tan relevante para este proyecto, es que estas cámaras no "ven" a través de los objetos, sino que recogen la emisión de estos. De esta forma, no pueden obtener la información que exista más allá de elementos transparentes o translúcidos, como sí harían las cámaras convencionales o las nocturnas.

Características y configuraciones de la cámara

Existe una amplia gama de cámaras infrarrojas, pero en concreto el modelo a utilizar es FLIR Indigo Omega. Para obtener un correcto análisis del entorno es necesario que la cámara está adecuada a él, y sea capaz de captar todo lo necesario. Es por ello que hay algunas características que nos permitirán obtener mejores resultados y configurar así nuestra cámara.

Una de ellas es la resolución. Estos dispositivos no obtienen imágenes muy grandes, pero si lo suficiente para futuros procesados y análisis. Otra característica sería el color bajo el que trabajamos. Suelen contar con opciones de escala de grises, como en nuestro caso, arco iris o *iron*. Esto puede verse en la figura 3.11, donde el ejemplo de *iron*, corresponde con la imagen de la derecha. Dependiendo de la aplicación, será conveniente utilizar unos u otros. Así, la escala de grises permite resaltar detalles en la imagen, lo cual es interesante para nuestra aplicación, así como la elección de arco iris tiene mejor sensibilidad para mostrar las diferentes temperaturas. La sensibilidad será otro parámetro relevante puesto podremos saber cuál será la diferencia mínima captada por esta, así como el rango de temperaturas, el cual para nuestra cámara esta entre -20ºC y 1200ºC.

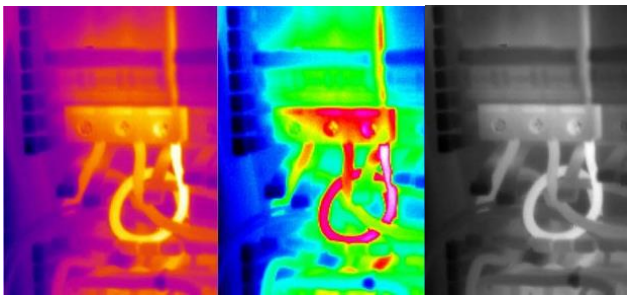


Figura 3.11: Paleta de colores [22]

Además de estas características, para realizar buenas capturas y tener muestras que nos permitan extraer todos los detalles posibles, es necesario tener en cuenta otros factores como el enfoque, el cual es el más relevante para resultados útiles, ya que una vez obtenidas las capturas, modificar el enfoque desde un ordenador será ineficaz e improductivo; o la temperatura reflejada, que nos permitirá compensar la temperatura reflejada por objetos del entorno. Este, junto a la emisividad, que es la proporción de radiación emitida por una superficie comparada con su temperatura, son parámetros a modificar, especialmente, si se comparan dos objetos

3.1.2 Métodos de detección

Tanto en el paso de la toma de imágenes como en el posterior análisis de estas, existen distintos métodos que nos ayudarán a precisar nuestra búsqueda y conseguir un mejor

detector. Como se ha mencionado en el apartado de Estado del Arte, podemos diferenciar distintos métodos según se coloquen las cámaras, por o según se obtenga la información de las imágenes y se cree el algoritmo que nos permitirá las detecciones.

3.1.2.1 Método según el número de fuentes

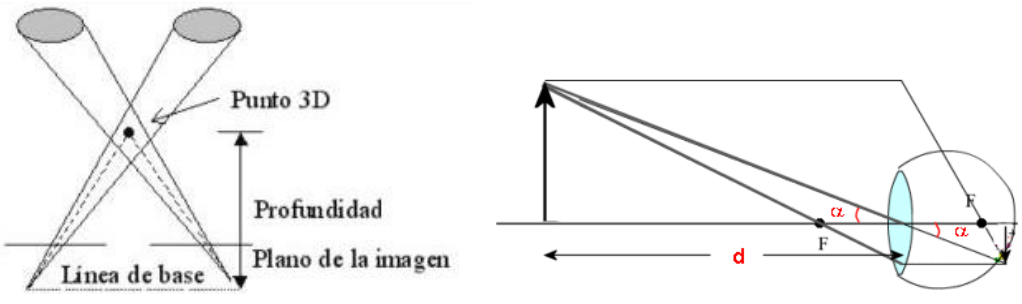
Existen distintos métodos para la detección de vehículos, y objetos en general. Podemos diferenciar entre:

Métodos basados en visión estéreo

La visión estéreo es aquella que trata de obtener la máxima información posible a través de varios dispositivos, de forma que se amplíe el rango visual. A la hora de procesar la información, este método hace mucho más fiable la información obtenida, puesto que en la mayoría de las ocasiones, se suelen superponer y comparar gracias a la adquisición de los distintos aparatos.

Como se ha comentado antes, en la obtención de las imágenes en algunas ocasiones, se distorsiona la realidad. Uno de los casos con el cual nos podemos encontrar es por ejemplo en la transformación del 3D al 2D, hace que mucha información se pierda por el cambio de dimensión. De esta forma nacieron los sistemas en estéreo, con la intención de, al utilizar dos o más imágenes del entorno, ser capaces de recrear esa profundidad perdida y dar la correcta forma a los objetos. A partir de esto, se puede determinar la distancia correcta que existe al objeto en cuestión, y en caso de nuestro proyecto, al vehículo para evitar la colisión.

Para poder captar esto se recurre al uso de dos cámaras situadas a la misma altura y separadas horizontalmente de forma que sean capaces de recrear lo que el sistema de visión humano podría ver en este momento. Esto se podría representar como el diagrama de la figura 3.12a. A diferencia del ser humano, este sistema de visión se basa en ejes paralelos (figura 3.12b), como se ha mencionado anteriormente en la parte de óptica, frente a los ejes convergentes del sistema humano. Las geometrías convergentes hacen muy complicado el cómputo de las imágenes, lo que con los ejes paralelos se solventa eficazmente sin perder información.

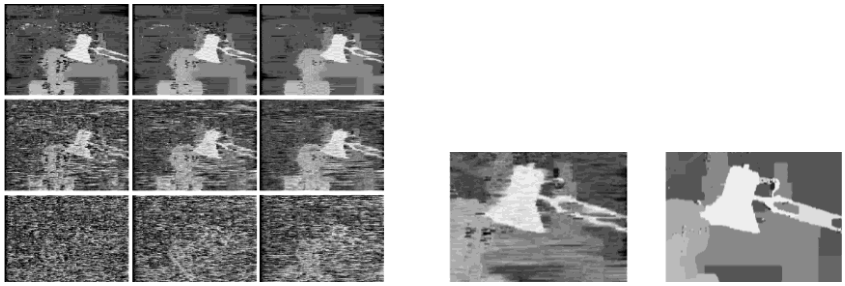


(a) (b)

Figura 3.12: (a) Visión en estéreo [23] (b) Óptica del ojo humano [24]

Sin embargo, el hecho de tratar con distintos dispositivos también puede ser un inconveniente. En el caso de las cámaras por ejemplo, la calibración de estas es un reto difícil, así como la sincronización de las imágenes, debido a la inestabilidad o a la vibración de estas cuando los vehículos circulan a una velocidad elevada. A pesar de ello, gracias a las mejoras que estos sistemas han sufrido, siguen siendo una de las mejores opciones para la detección de objetos en vehículos móviles.

Algunas aplicaciones que utilizan este tipo de sistemas son por ejemplo los mapas de disparidad, los cuales pueden definirse como el conjunto de píxeles distintos de diferentes imágenes de una misma escena. Gracias a estos se pueden reconstruir los mapas en 3D, siempre y cuando se conozcan los parámetros de la plataforma donde se encuentra el sistema estéreo, como es el caso del ejemplo de las figuras 3.13a y 3.13b.



(a) (b)

Figura 3.13: (a) Mapas de disparidad [25] (b) Recreación modelos 3D [25]

Métodos basados en visión monocular

Los sistemas monoculares, los basados únicamente en una fuente de información, extrapolándolo al ejemplo anterior, lo que sería una única cámara, a pesar de contar con la desventaja de no poder recrear el entorno exactamente como las estéreo, son capaces de obtener resultados ampliamente aceptables, y siguen siendo utilizados, como es el caso del vehículo autónomo con el que se ha trabajado. Este sigue modelos como el *pinhole* o el de lente fina explicados anteriormente.

3.1.2.2 Métodos basados en el movimiento

La detección del movimiento es un tema fundamental en la visión artificial. Este tipo de modelos basan su búsqueda de posibles vehículos no por la posición de estos o las características que presentan, sino por los movimientos que estos pueden estar realizando en la imagen. Así, a partir de ellos, somos capaces de averiguar si estamos ante un caso de vehículo o no. De esta forma, el fundamento del método es calcular el movimiento en 3D de los objetos, lo que nos permite estudiar el flujo óptico de la secuencia de imágenes, gracias al cual, se crea un mapa con los vectores de movimiento. El flujo óptico es el movimiento de los pixeles de una imagen a otra. Su cálculo no es sencillo, y en la mayoría de las ocasiones, está basado en aproximaciones

o hipótesis que en la vida real no se cumplen. Un ejemplo de flujo óptico para esta aplicación, podría ser la figura 3.14.

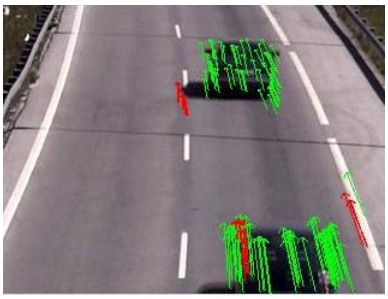


Figura 3.14: Detección de vehículos por flujo óptico [26]

Dependiendo de cómo se detecte el movimiento tendremos dos opciones de flujo, denso o disperso. Esto hace referencia a qué imágenes tomamos para calcular ese movimiento.

- Flujo óptico denso: se tiende a estudiar todos los puntos de las imágenes y así extraer las características.

- Flujo óptico disperso: por el contrario, este utiliza solo algunos puntos para la extracción de información, siendo estos los relevantes a nuestra búsqueda.

Para calcular estos flujos ópticos será necesario la obtención de las imágenes y los puntos de estas. Si un punto se ha movido de una imagen a otra, esto implica que su flujo óptico ha cambiado. Estos se estiman partir de las derivadas espacio-temporales, lo que supone una complejidad en la aplicación. Debido a esa dificultad y a las aproximaciones que se realizan, para mejores resultados, el flujo óptico se agrupa para eliminar los valores atípicos que puedan existir.

Por lo general, para este tipo de aplicaciones destinadas a tiempo real, el estudio del flujo óptico denso resulta un método muy lento para las detecciones. Es por ello que, para este tipo de aplicaciones, es más recomendable el modelo disperso, puesto que sólo se centra en algunos pixeles concretos. Sin embargo, en general, el método de detección por movimiento no es muy utilizado para estas aplicaciones, puesto que no cumplen con el objetivo temporal descrito.

3.1.2.3 Métodos basados en la apariencia

Para los sistemas de detección, gracias a los avances de la visión artificial, la apariencia está teniendo cada vez más importancia, puesto que gracias a los distintos métodos existentes, el reconocimiento de objetos es mucho más sencillo y rápido que en métodos como el de detección de movimiento o el de extracciones de características mediante

transformaciones en la imagen.

Como se ha mencionado anteriormente, en esta clasificación podemos diferenciar entre los métodos basados en modelos o los basados en aprendizaje, aunque el fundamento de ambos es la extracción de características, eso sí, esta vez relacionadas con la apariencia del objeto. Sin embargo, a pesar de partir de lo mismo, son dos métodos cuyos resultados difieren bastante. El método basado en modelos no es tan preciso como puede ser un método de aprendizaje, ya que, por ejemplo en el caso de los vehículos, debería contar con una gran base de datos de la que poder extraer todas las características de cada vehículo, dado que estos, a pesar de presentar unas similitudes en tamaño y forma, siguen contando con una amplia gama de características no comunes, que son las que hacen que el método de los modelos no sea del todo fiable para esta aplicación.

Métodos basados en modelos

Estos varían según la aplicación que tengan o el objeto que se esté estudiando, ya que en los últimos años de investigación de la visión artificial, las técnicas han avanzado notablemente, y se han realizado numerosas aplicaciones para distintos obstáculos, conociendo ahora cuales son las características principales de los objetos.

Los métodos basados en modelos, o plantillas, centran su búsqueda encontrar el mayor número de semejanzas en la imagen en cuestión, con respecto a la plantilla o modelo concreto. El principal inconveniente es el que ya se ha mencionado, es necesario una gran base de datos para la correcta clasificación de los vehículos que puedan circular por una vía, puesto que de la otra forma, tendríamos numerosos falsos negativos que no serán reconocidos.

Un método basado en modelos, además del que recorre las plantillas por la imagen a examinar, puede ser la búsqueda de vehículos por su forma cuadrangular, ya que se resume los coches en un par de líneas horizontales y verticales. Esto, junto con limitaciones geométricas de posiciones imposibles de los vehículos, por los tamaños de estos y la distancia a la que se pueden encontrar, pueden dan lugar a un sistema de detección bastante riguroso. De igual manera ocurre con la mayoría de los sistemas de detección de matrículas, los cuales basan sus análisis en la búsqueda de líneas verticales.

Métodos basados en aprendizaje

Una de las ramas de la inteligencia artificial es el denominado aprendizaje automático o *Machine Learning*, el cual pude ser vulgarmente definido como el aprendizaje de las

máquinas. Según Arthur L. Samuel, el cual fue un pionero americano en el campo de los juegos de ordenador, así como la inteligencia artificial o en aprendizaje automático: "el aprendizaje automático es el proceso que le da a las computadoras la habilidad de aprender sin ser explícitamente programadas".

Llevado a la práctica, puede definirse como la creación de un programa que pueda aprender a realizar una tarea o procedimiento y poder mejorarse a sí mismo a través de su propia experiencia. Este concepto todavía es relativamente nuevo para este campo, ya que además, con los avances de las nuevas tecnologías, las investigaciones evolucionan muy rápido, quedándose obsoletos métodos de tan solo unas décadas atrás.

Estos procedimientos, fundamentados en una base de datos de imágenes, son capaces de clasificar o agrupar los elementos como objeto o no-objeto (en este caso vehículo o no-vehículo), así como proporcionar algún tipo de valor numérico al respecto, en algunos casos, como será comentado posteriormente. A partir de las imágenes, se extraen las características, normalmente en forma de vector, que serán utilizadas para el posterior entrenamiento de este clasificador. Este tipo de métodos cuentan con una desventaja muy importante, y es que para conseguir un sistema lo suficientemente robusto, es necesario una gran base de datos, y una cuantiosa variabilidad del propio objeto. Así por ejemplo, para el caso de los vehículos, ha sido necesario incluir un considerable número de modelos de coches para que el detector fuese capaz de reconocerlos.

Así, en este tipo de métodos podemos diferenciar el aprendizaje supervisado, el cual durante el entrenamiento conoce cuál es el objetivo de la detección, y el aprendizaje no supervisado, que al contrario que el otro, no cuenta con conocimiento previo de cómo debe ser el objeto a detectar. De forma que se tenga una mejor comprensión de los conceptos de aprendizaje supervisado y no supervisado, vamos a extrapolarlo a un ejemplo muy sencillo:

Imaginemos que tenemos una cesta de fruta en la que tenemos distintos tipos de fruta y queremos separarlas según la fruta que sea.

Si esto se quisiese hacer mediante un entrenamiento supervisado, nos basaríamos en lo que sabemos de cada fruta. Con previo conocimiento de ellas, sabemos que se pueden diferenciar, por ejemplo, por estructura. De esta forma, en base a nuestros conocimientos, partiríamos de unas características buscadas que permitirían que el aprendizaje esté direccionado a ese camino concreto. Así, el detector aprenderá a reconocer las frutar por su forma.

Sin embargo, en el caso del entrenamiento no supervisado partimos únicamente del cesto de fruta, pero ningún conocimiento añadido. De esta forma el detector, durante el entrenamiento encontrará características comunes a los elementos que le presentamos y

él mismo será capaz de distinguir unas piezas de otras. Así por ejemplo, puede partir agrupando la fruta por colores, dejando así pequeños grupos similares. Tras esto, realizará otro entrenamiento en el que resalte otra característica y así sucesivamente hasta tener diferenciadas todas las piezas.

Partiendo de esa base, podemos diferenciar algunos métodos conocidos como:

- Deep Learning: sobre el cual no existe una definición clara, pero puede denominarse como el conjunto de algoritmos, tanto de aprendizaje supervisado, como no supervisado, basado en el aprendizaje a múltiples niveles de redes neuronales, de procesamiento no lineal, que permite la creación de sistemas de detección, reconocimiento o modelación de datos. Este método, relativamente moderno, está revolucionando las técnicas hasta ahora utilizadas en la visión artificial, y está ganándoles terreno.

- K-means Clustering: método de aprendizaje no supervisado y está basado en el valor medio asignado a cada observación. Estas serán clasificadas según el grupo más cercano con un valor medio similar. Es muy utilizado, sobre todo para diferenciar objetos en una escena.

- Support Vector Machines: basado en el aprendizaje supervisado, es un método usado para la clasificación y la regresión. Basado en un conjunto de imágenes de entrenamiento, se construye un modelo para que permita la detección del objeto en cuestión en una nueva muestra. Gracias a una investigación de Dalal y Triggs [3], que se explicará posteriormente, se descubrió que el entrenamiento con SVM basándose en las características extraídas por Histogramas de Vectores Orientados, proporcionaba unos resultados espectaculares, comparados con métodos usados hasta entonces.

- Adaptative Boosting: entre los cuales se encuentra el famoso método de Viola Jones [2], Haar Cascade, es un método de aprendizaje supervisado que, mediante la iteración, consigue tras varias etapas, las cuales actúan en forma de cascada aprendiendo de lo recién aprendido, que se obtenga el clasificador final.

Este trabajo se ha centrado en los dos últimos métodos de aprendizaje supervisado, los cuales serán explicados más detenidamente. Así, intentaremos obtener el mejor sistema de detección basado en visón monocular con una cámara infrarroja obtenida a partir de los métodos de aprendizaje supervisado, SVM junto con HOG y las cascadas de Haar, para poder conseguir la correcta clasificación de vehículo y no-vehículo.

3.2 Sistemas de aprendizaje supervisado

Como ya se ha comentado, el aprendizaje supervisado es uno de los tipos de *Machine Learning* que a día de hoy se conocen. Se podría considerar como uno de los más claros a ejemplificar y de entender, puesto que en base a lo que se le introduce, así reacciona, aprendiendo de una serie de características, al contrario que el no supervisado.

El método de aprendizaje supervisado trata de obtener un método o una regla gracias a la cual es capaz de obtener respuesta para los objetos que se nos presenten, lo que en nuestro caso se equipará a la detección de vehículos. Debe, a partir del conjunto inicial de vehículos que le introducimos, deducir un sistema que sea capaz de seguir detectando futuros automóviles.

Matemáticamente esto puede corresponder a:

$$\boldsymbol{h(x) = X \rightarrow Y} \tag{3.4}$$

$$siendo:$$
$$X: conjunto\ objetos$$
$$Y: conjunto\ respuestas\ (vehiculo, no - vehiculo)$$
$$h(x) = función\ entre\ X\ e\ Y$$

Esta función, lo que permite, es lograr que toda máquina basada en ello sea capaz de predecir el valor correspondiente a cada objeto sin importar la entrada introducida, gracias a unos datos de entrenamiento iniciales. Con el fin de que esto sea posible, es necesario obtener todos esos elementos imprescindibles para el aprendizaje.

De esta forma, a partir de los objetos que van a resultar como modelo para el entrenamiento, obtenemos las distintas características en las que nos podemos basar. Estas características, también denominadas *features*, serán la información necesaria para comenzar nuestros entrenamientos. De manera matemática, se podría entender como un vector del objeto que recoge las distintas características de este. Así por ejemplo, en el caso de los vehículos, cada vehículo de la base de entrenamiento contará con algunas características como el tamaño, la forma o la disposición de algún elemento. Esta información será extraída de maneras distintas, como por ejemplo en el método de los HOG, este proceso es el encargado de obtener esas características de cada imagen que es introducida, obteniendo al final una serie de vectores de gran relevancia.

El conjunto de respuestas, por otro lado, es algo distinto a lo mencionado. Los métodos de aprendizaje supervisado pueden utilizarse para clasificación, como es el caso de vehículo sí o vehículo no, o para regresión, en la cual ya intervienen más factores, y las respuestas a una detección no serían propias de una clasificación, como vehículo o no-vehículo, sino que describiría en cierta medida alguna característica, generalmente de forma numérica. De esta forma, si quisiésemos adaptar nuestro algoritmo a una

regresión, esta podría indicarnos por ejemplo, cómo de grande o de pequeño es el vehículo que se está detectando. Sin embargo, para este proyecto sólo nos vamos a centrar en la función de clasificador.

Así, una vez obtenida la información necesaria, pasaríamos al entrenamiento, que nos proporcionaría una dependencia entre los objetos y la respuesta que estamos buscando. Esta función, es lo que se denomina hipótesis. Una vez obtenida, es necesario comprobar que la hipótesis nos da los resultados deseados. En el caso de no ser así, se podrían corregir algunos parámetros utilizados durante en el entrenamiento, o utilizar esas muestras erróneas para volver a entrenar la hipótesis. La ventaja de este tipo de procedimiento es, que es posible observar que está clasificando el detector. Al ser un aprendizaje supervisado, podemos corregir aquello que no funcione correctamente, pudiendo observar claramente los efectos que los distintos parámetros tienen sobre el sistema, mientras que para el no supervisado, es muy complicado encontrar el problema que no permite una buena clasificación.

De esta forma, pasaremos a explicar los dos métodos, basados en aprendizaje supervisado, utilizados para la realización de este proyecto. Cabe destacar que son métodos suficientemente distintos, aunque sí que comparten cosas en común, como es por ejemplo las técnicas utilizadas para la extracción de características o para la búsqueda de objetos durante los entrenamientos. A pesar que el primer método utiliza el descriptor HOG para obtener las características y el segundo el de Haar, ambos hacen uso de técnicas como la de la ventana deslizante. Esta permite recorrer las imágenes con una ventana de un tamaño prefijado, con el fin de, en cada movimiento, obtener o extraer información de esa ventana en cuestión, y obtener lo relevante de ella.

3.2.1 Método de Dalal y Triggs

En 2005, un grupo de investigadores sacaron a la luz un método, que desde entonces, ha sido una gran base para los sistemas de detección de objetos, así como para la visión artificial en general. Estos investigadores, Navneet Dalal y Bill Triggs crearon un sistema de detección de peatones basado en la combinación de los Histogramas de Gradientes Orientados, HOG, y los Support Vector Machine, SVM, [3].

Después de numerosas investigaciones y estudios, llegaron a la conclusión que, para la aplicación bajo la que estaban trabajando, el descriptor HOG, era capaz de conseguir mejores resultados que los descriptores utilizados hasta entonces. Este, gracias a los gradientes, remarcaba ciertas características de los peatones, como su forma o estructura, que ayudaba a diferenciarlos de otros objetos, permitiendo además que esa información tan relevante, quedase recogida en el descriptor final. Descubrieron que una de las características más importantes de HOG, la cual es la normalización por contraste local, era vital para un buen resultado.

Este trabajo, en un principio se centró en la correcta división de una base de datos de peatones, ya existente en la red, procedente del MIT. Sin embargo, al obtener tan buenos resultados, se decidió ampliar esa base de datos con nuevas imágenes, así como cambios de fondo y posturas de los peatones. De esta forma, no sólo consiguieron la detección de peatones en entornos sencillos o en posturas verticales, sino que además consiguieron grandes resultados en la detección de peatones que se podían encontrar andando o realizando alguna otra tarea. La diferencia principal con respecto a otros métodos, fue la conclusión que las formas o contornos de los objetos pueden ser aceptablemente caracterizadas por los gradientes de la imagen. Su procedimiento fue el siguiente:

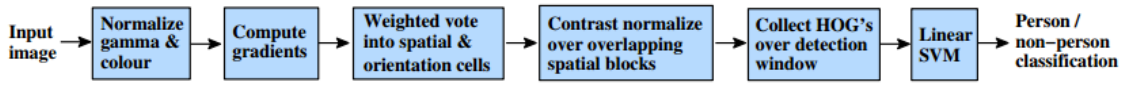


Figura 3.15: Esquema de la investigación [3]

Tras conseguir las imágenes de muestra o de entrenamiento, estas se procesaban para obtener una mejor visualización de sus características. Acto seguido, se calculaban los gradientes de estas nuevas imágenes y, según unos pesos proporciones, los cuales se comentarán más adelante, se calculaban los histogramas. Posteriormente, de forma proporcional, nuevamente, se obtenían los vectores de características pertenecientes a cada bloque de la imagen, y por último, se computaba el vector final. Este sería introducido en un SVM lineal, y se obtendría el clasificador de persona o no-persona.

Los resultados obtenidos fueron realmente eficaces, lo que consiguió una llamada de atención al campo de la visión artificial a la aplicación de este método en otros estudios. Desde entonces, esto se ha utilizado, no sólo para la detección de peatones, el cual ya quedó demostrado que era ideal para ello, pero también para otros objetos como señales de tráfico, ciclistas o vehículos, como es el caso de este proyecto.

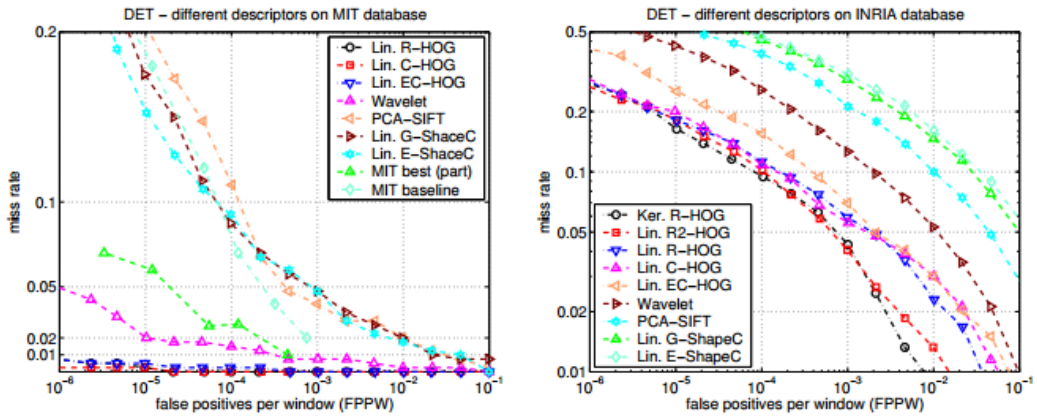


Figura 3.16: Curvas Trade-off [3]

Como se ha visto, para el desarrollo del clasificador final, es necesario hacer uso de dos herramientas: un descriptor, el cual recogerá las características de las imágenes que introduzcamos como fuente de información; y lo que se conoce como método de clasificación, que nos permite designar la frontera, la cual es la línea que nos permite diferenciar los distintos grupos que tenemos en nuestra imagen. Para este método, las herramientas utilizadas fueron el descriptor de Histogramas de Gradientes Orientados y el Support Vector Machine, respectivamente. Estos serán ampliamente descritos a continuación con el fin de entender el procedimiento completo.

3.2.1.1 Histogramas de Gradientes Orientados (HOG)

Los gradientes nos informan de los cambios de intensidad de una imagen, y para cada pixel, este tiene una magnitud y una orientación, donde ese cambio de magnitud es máximo. Los gradientes nos permiten obtener información muy relevante sobre la forma del objeto, ya que, por lo general los contornos de los elementos tienen gradientes de magnitudes más elevadas. La orientación, por su parte, nos proporciona información sobre la forma del contorno, mientras que la magnitud nos indica como de importante es el cambio de intensidad en la imagen.

De esta forma, para calcular el gradiente en un pixel, es necesario recurrir a sus pixeles vecinos, puesto que según el concepto de gradiente, es imprescindible conocer cómo cambia la intensidad de unos con respecto a los otros. El gradiente se calcula tanto en el *eje x* como en el *eje y*, y para el resultado final, se unen o solapan ambos ejes. De forma matemática, para el cálculo de los gradientes en las distintas direcciones recurrimos a:

$$\boldsymbol{dx = I(x+1, y) - I(x-1, y), para\ el\ eje\ x} \tag{3.5}$$

$$\boldsymbol{dy = I(x, y+1) - I(x, y-1), para\ el\ eje\ y} \tag{3.6}$$

Desde aquí, se pueden representar estos valores como parte de un vector, $dv = (dx, dy)$ y de esta forma poder calcular la orientación y la magnitud, como se muestra en la figura 3.17.

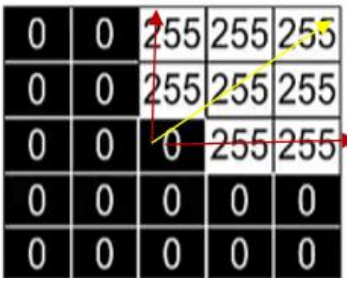


Figura 3.17: Vector de gradiente

Para el cálculo de la orientación y de la magnitud, basta con recurrir a las relaciones trigonométricas en la imagen anterior, obteniendo:

$$\boldsymbol{\theta(x,y) = arctan\left(\frac{dy}{dx}\right)} \quad (3.7)$$

$$\boldsymbol{g(x,y) = \sqrt{dx^2 + dy^2}} \quad (3.8)$$

Sin embargo, para que esta información sea utilizada por un clasificador, es necesario que se obtenga una representación global de toda la imagen y no de pixel a pixel, puesto que tal y como están orientados los clasificadores, no es posible hacer uso de ello. Además, al utilizar la información de cada pixel, condicionamos el resultado final a cada pequeña variación que estos, individualmente, puedan sufrir. De esta forma, lo que los Histogramas de Gradientes Orientados persiguen, es transformar toda esa información local de cada pixel, a una estructura global que te proporcione las características generales del objeto en forma un descriptor que pueda ser introducido, posteriormente, en un clasificador.

Con el fin de encontrar esa representación global de la imagen, el descriptor HOG sigue dos pasos fundamentales:

- División de la imagen y obtención de los histogramas: La imagen es dividida en un número fijo de celdas, para cada una de las cuales se calcula el Histograma de Gradientes Orientados. Para calcularlos es necesario tener en cuenta una serie de parámetros claramente relevantes en el resultado final. Entre ellos podemos destacar:
    - Tamaño de la celda: es el tamaño de las celdas en las que se dividirá la imagen, como se puede observar en la figura 3.18. Los valores recomendados para óptimos resultados suelen ser entre 4 y 8 pixeles, tanto en ancho como en alto, puesto que se tiende a desplazarse con ventanas cuadradas. Cada celda de esta ventana se denominará subcelda. Cuanto mayor sea el tamaño de la ventana, menos precisión se obtendrá, pero el gasto de cómputo será menor también.

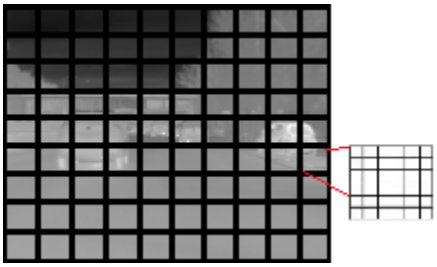


Figura 3.18: Tamaño de la celda

- Orientación: la cual estará designada por dos indicadores. El primero nos ayuda a unificar los gradientes, agrupando así a aquellos que tienen una dirección y sentido similares. Es lo que se reconoce como intervalo de orientaciones y no es más que el número de grupos en los que dividimos nuestro rango de orientaciones. Un valor común para este parámetro está recogido entre 20-25º.

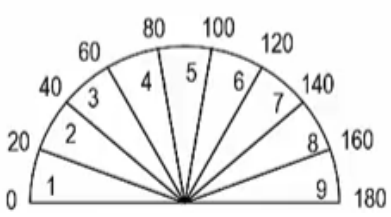


Figura 3.19: Orientaciones [14]

El segundo es ese rango que acabamos de mencionar. Según este sea, podremos considerar signo a los gradientes o no. En caso de tener una orientación de 0º a 360º, supondremos la orientación del gradiente sin signo. Sin embargo, en el supuesto de tener un rango de 0º a 180º, como es el caso de la imagen 3.19, ya si será necesario trabajar con signos, y los gradientes con la misma dirección pero de sentido opuesto, quedaran recogidos en el mismo intervalo de orientación.

Una vez definidos los parámetros, los gradientes quedarán controlados por las pautas recién explicadas. De esta forma, para cada subcelda, el gradiente correspondiente, designado por magnitud y orientación, quedará recogido o se comportará según el intervalo correspondiente le dicte, obteniendo así grupos de gradientes pertenecientes a los mismos intervalos, según la figura 3.20.

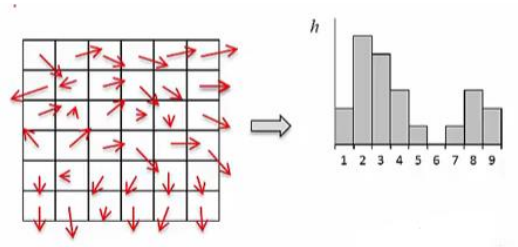


Figura 3.20: Generación del histograma de la subcelda [14]

Por último, y como paso final para el cálculo de los histogramas de cada celda inicial en la que se dividió de la imagen, será calculado sumando el número de gradientes pertenecientes a cada intervalo. Matemáticamente esto quedaría recogido de la siguiente manera:

Siendo C, una celda con un conjunto de pixeles (x, y) , los cuales quedan definidos por su magnitud y su orientación

($\theta(x,y)$ y $g(x,y)$ respectivamente), con el fin de calcular el valor del histograma en una posición k, este valor es calculado como la suma de la magnitud de los gradientes para todos los pixeles de la celda ponderados por una factor de relación entre el gradiente y el intervalo correspondiente.

$$h(k) = \sum_{(x,y)\in C} \omega_k\,(x,y) g(x,y) \tag{3.9}$$

En sus primeros estudios, el valor de ponderación tomaba valor 1 o 0, dependiendo de si el gradiente correspondía al intervalo a estudiar o no, pero sin embargo, esto generaba algunos problemas de precisión final. Aquellos gradientes que se encontrasen en los límites de los intervalos, cambiaban significantemente los resultados al hacer esa aproximación. De esta forma, para corregir ese problema, el valor de parámetro de ponderación pasó a ser un peso proporcional a la distancia entre esos gradientes y aquellos situados en el centro de los intervalos. La relación matemática que lo define es la siguiente:

$$\boldsymbol{\omega_k(x,y) = max\left(0, 1 - \frac{\theta(x,y)-\theta_k}{\delta\theta}\right)} \tag{3.10}$$

Este cambio permite que los gradientes de cada subcelda no afecten únicamente a un intervalo, si no a los dos que lo rodean, consiguiente así un entorno global final más preciso. Esta puede verse representada en la figura 3.21.

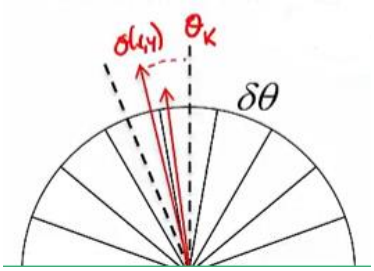


Figura 3.21: Interpolación espacial [14]

Como se ha comentado, este procedimiento se realiza para todas las celdas de la imagen, de forma que para cada una de ellas, obtenemos el histograma correspondiente. Sin embargo, puede ocurrir que como con los gradientes que se encuentran cercanos a los límites de los intervalos, existan pixeles en los límites de las celdas, que su pertenencia a una o a otra de ellas, tenga repercusiones significativas en los resultados. Los píxeles que pueden pertenecer al mismo contorno, pueden quedar divididos en celdas diferentes.

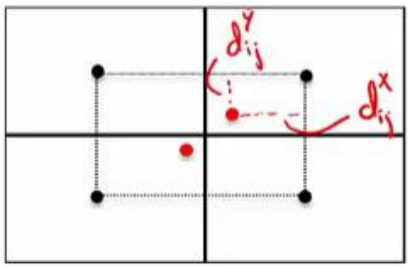


Figura 3.22: Interpolación espacial [14]

Para solucionar este problema, se recurre al mismo principio anterior. Cada pixel es utilizado en el cálculo de histogramas de la celda a la que pertenece y las otras tres más cercanas, con un peso proporcional a la distancia entre el pixel en cuestión y el centro de cada celda, como se percibe en la figura 3.22, de forma que se añade más información a los histogramas locales, y por lo tanto, también al global. Matemáticamente, esto queda recogido:

$$\boldsymbol{h_{ij}} = \sum_{(x,y)} \boldsymbol{\omega_{ij}^{x}(x,y)}\, \boldsymbol{\omega_{ij}^{y}(x,y)\omega_{k}(x,y)g(x,y)} \tag{3.11}$$

Teniendo en cuenta tanto la distancia en el eje x como en el eje y para este cálculo.

- Cálculo del descriptor global: gracias a la combinación de todos los histogramas de cada una de las celdas obtenidos en el procedimiento anterior, es posible calcular el descriptor general que recoge el comportamiento de la imagen. Se combinan todos los histogramas obtenidos con el fin de ofrecer la información global en base a un único vector, información que corresponde a los contornos y los bordes del objeto. Para obtener el descriptor final, será necesario normalizar y unificar los histogramas de cada celda.

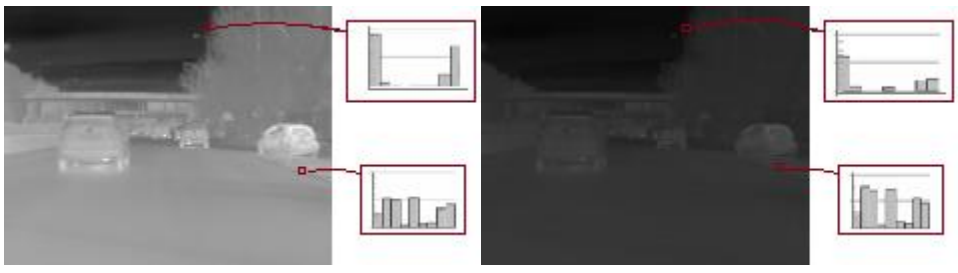


Figura 3.23: Variación gradientes ante cambios de iluminación, sin normalización

Con el fin de obtener un buen descriptor, y por lo tanto un preciso clasificador, es necesario que este sea invariante a los cambios que puedan existir en la imagen, como por ejemplo los cambios de iluminación o la diferencia de contrastes, para evitar casos como el representado en la figura 3.23. Estas

modificaciones para conseguir la invarianza en la imagen, ya se han tenido en cuenta, en menor medida, en el apartado anterior, al incluir el gradiente resultante de cada pixel en varias agrupaciones.

Con el objetivo de unificar la magnitud global del gradiente de todas las imágenes, incluso cuando estas están afectadas por cambios de iluminación o gran contraste, es necesario normalizar los valores de los histogramas. Esto permite que una misma imagen con distintas iluminaciones tenga como resultado la misma representación de histogramas, como ocurre en la figura 3.24, la cual, aunque sigue presentando diferencias, no son tan relevantes.

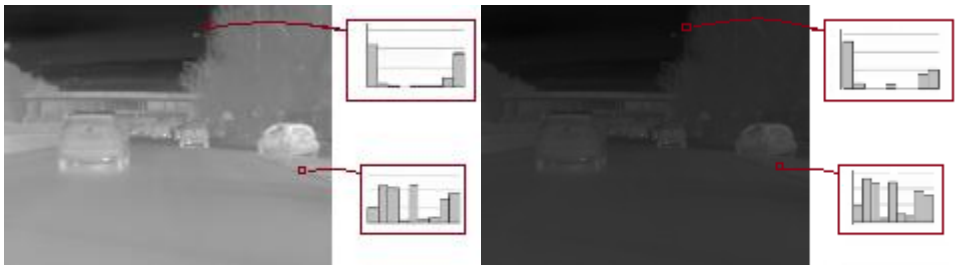


Figura 3.24: Variación gradientes ante cambios de iluminación, con normalización

Muchos procedimientos de visión por computador, para normalizar los valores de los pixeles de una imagen, por ejemplo, aplican un mismo Kernel a todos los pixeles y cambian sus valores respectivamente. Sin embargo esto no puede ser ejecutado de igual forma en este caso. Los cambios de intensidad no son iguales en cada celda de la imagen y es por ello que no todas se merecen el mismo comportamiento. De esta forma, con el fin de hacerlo de la manera más suave, nació el concepto de bloque, que no es más que un conjunto de celdas vecinas. Este puede apreciarse en el rectángulo rojo de la imagen 3.25.

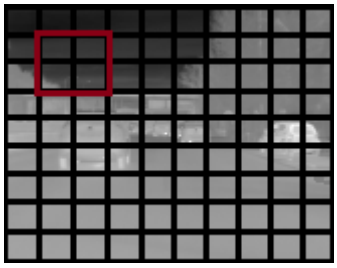


Figura 3.25: Concepto de bloque

Para realizar la normalización, se toma un bloque de un tamaño determinado, siendo el recomendado de 2 celdas tanto en vertical como en horizontal, y se toman los histogramas de esas celdas afectadas. Estos se concatenan con el fin de formar un vector donde se recojan todas las características de ese bloque. Ese vector será el que se normalizará, para conseguir esa invarianza, y el que se

utilizará para calcular el descriptor HOG final. La normalización se consigue dividiendo cada elemento del vector por el módulo, o norma, de éste. El módulo de un vector viene dado por:

$$\|\boldsymbol{v}\| = \sqrt{\sum \boldsymbol{x_i^2}} \tag{3.12}$$

Por lo que la normalización del vector vendrá definida por:

$$\boldsymbol{v'} = \frac{\boldsymbol{v}}{\sqrt{\|\boldsymbol{v}\| + \boldsymbol{\varepsilon}}} \tag{3.13}$$

De esta forma, al dividir por el módulo, garantizamos que la magnitud total del vector es la unidad, y que además las pequeñas diferencias de contraste en la imagen ya no tienen tanta repercusión en el resultado final. Para aquellas zonas donde la diferencia de intensidad no exista, y por lo tanto el gradiente sea nulo, y con el fin de evitar tener una división por cero, se ha añadido la constante ε. Esta toma un valor muy pequeño, por lo que en condiciones de división por un numero distinto a cero, la constante no es notable.

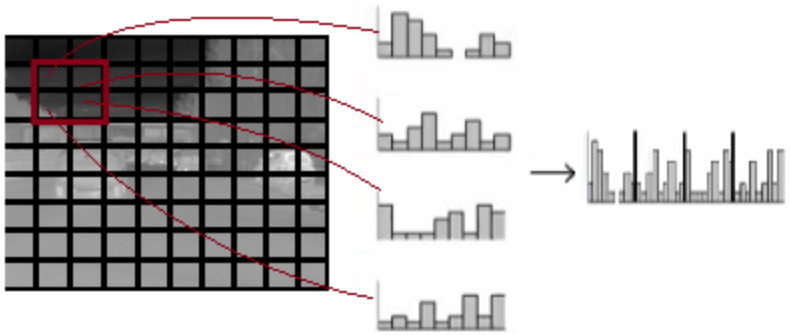


Figura 3.26: Proceso de normalización

En la figura 3.26 anterior podríamos ver el resultado de esa normalización, con bloques más pequeños. Claramente se puede observar que, aunque sigan existiendo diferencias en los histogramas, estás se han reducido notablemente, y lo más importante, la magnitud global del bloque es igual para ambas imágenes.

A partir de la información de los bloques, se construye el descriptor final. Sin embargo, suponiendo un desplazamiento entre bloques de una celda de unidad tanto en horizontal como en vertical, podemos observar que la información se irá solapando, puesto que una misma celda podrá ser utilizada hasta en cuatro bloques, como se muestra en la imagen siguiente.

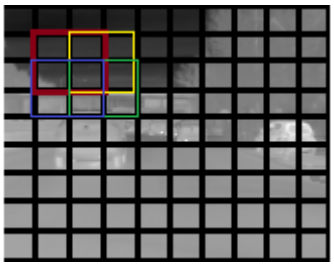


Figura 3.27: Solapamiento de bloques

Sin embargo esto lejos de ser un problema, es una ventaja para el descriptor, ya que de esta forma se está ofreciendo más información, y por lo tanto se está creando un descriptor más robusto e invariable. A pesar de ser el mismo histograma aplicado en cada bloque, al normalizarse en cada uno de ellos por un valor distinto, debido a los valores de las celdas vecinas, la contribución de la celda en cuestión a cada bloque será distinta. De esta forma, la obtención de este vector HOG viene dada por la concatenación de todos los vectores de histogramas obtenidos a partir de todos los bloques de la imagen. Este gran vector, se vuelve a normalizar, tal y como se ha explicado anteriormente, para mayor robustez e invariabilidad. La figura 3.28 recoge este proceso.

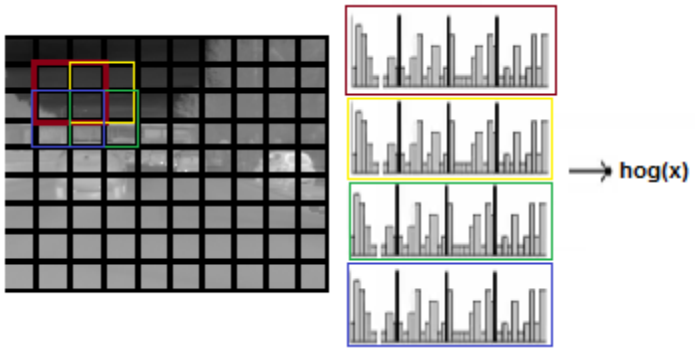


Figura 3.28: Generación del descriptor HOG

A modo de resumen, podemos observar como los distintos parámetros con lo que se ha ido trabajando, el tamaño de la celda, el signo del gradiente, el número de intervalos y el número de celdas en cada bloque, es representativo en el resultado final. De esta forma, con todos ellos podremos calcular cual es el número total de elementos en el vector HOG final, el cual es un parámetro usado como se comentará posteriormente en la parte de los entrenamientos. Este número viene dado por:

$$n = n^{\underline{o}}_{bloques} \cdot n^{\underline{o}}_{celdas/bloques} \cdot n^{\underline{o}}_{intervalos} \tag{3.14}$$

Donde el número de bloques de la imagen puede ser calculado gracias a:

$$n^{o}_{bloques} = n^{o}_{celdas} - n^{o}_{celdas/bloques} + 1 \quad (3.15)$$

3.2.1.2 Support Vector Machines (SVM)

Los Support Vector Machine o máquinas de vector soporte son unos de los nuevos métodos de clasificación. Imaginémonos un conjunto de piezas, de dos tipos distintos, dispuestos a lo largo de un plano horizontal. Imaginémonos que esta situación está vista desde una posición perpendicular a ese plano y podemos simular la transformación a un espacio de dos dimensiones. Una vez nos encontremos ahí, imaginémonos que queremos separar esos dos tipos de piezas mediante una línea. Si por ejemplo nos encontrásemos ante el caso de la imagen de abajo, sería una tarea sencilla, ya que los grupos pueden ser fácilmente diferenciados, y por lo tanto trazar esa línea no es ningún problema, como es el caso del siguiente ejemplo de la figura 3.29.

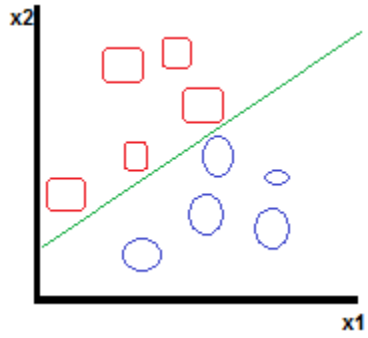


Figura 3.29: Clasificación de grupos linealmente separables

Sin embargo, aunque la solución inicial sería una línea que pase por el centro del margen entre ambos grupos, esta no es la única solución existente. Existen numerosas líneas capaces de realizar la tarea correctamente y es lo que se conoce como solución de hiperplanos. Esto quiere decir, que en este caso, teniendo dos grupos a separar y encontrándonos en un espacio de dos dimensiones, se pueden trazar numerosas líneas que separen los conjuntos basándose en criterios diferentes. Se pueden definir dos grandes caminos para poder hacer estas separaciones:

- Modelos generativos: los cuales se basan en la función de densidad de probabilidad asociadas a cada una de las clases. Este es el caso, por ejemplo, de los métodos de Bayes o de Fisher

- Modelos discriminativos: que al contrario que los anteriores, no utilizan estas fórmulas de densidad de probabilidad, sino que intentan encontrar la frontera, o esas líneas de división, mediante un conjunto de datos de entrenamiento, lo suficientemente representativo, del que aprenden a hacerlo. Este es el caso de los SVM.

Los SVM son sistemas de clasificación binarios, es decir, sólo pueden distinguir dos clases o conjuntos distintos. Son clasificadores lineales, lo que supone la solución de un SVM siempre va a ser un hiperplano que permita dividir el espacio en los dos conjuntos correspondientes. En el caso de un espacio de dos dimensiones, este hiperplano es una recta, como se puede comprobar en la figura 3.29 o 3.30, la cual nos muestra las posibles soluciones existentes; en uno de tres dimensiones nos encontramos con un plano; y en cuatro dimensiones o más estaríamos tratando con hiperplanos.

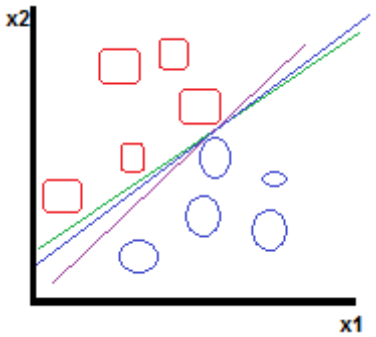


Figura 3.30: Conjunto de hiperplanos en 2D

Suponiendo de nuevo un espacio de dos dimensiones, para encontrar la línea que separa las clases, el SVM hace uso de unas muestras de entrenamiento de cada conjunto, que es lo que se llama vectores de soporte, para calcular la solución final. Estos vectores no son elegidos al azar, sino que tienen que cumplir una característica determinada. La distancia de separación entre los planos que contienen los vectores de soporte de cada conjunto, debe ser máxima. Con esto lo que se busca es obtener la zona del espacio entre vectores de soporte que se quede libre de muestras de los conjuntos, y que por lo tanto, recoge todas las posibles soluciones al problema en cuestión. Esto podría asemejarse a un problema matemático de optimización para encontrar el margen máximo entre dos conjuntos. Es por ello que la solución del SVM podría definirse como la región en el espacio más amplia, y libre de muestras, que permite la separación de los dos grupos que se pretenden separar. Esta queda recogida por línea negra, en la zona amarilla, que separa ambos conjuntos de la figura 3.31.

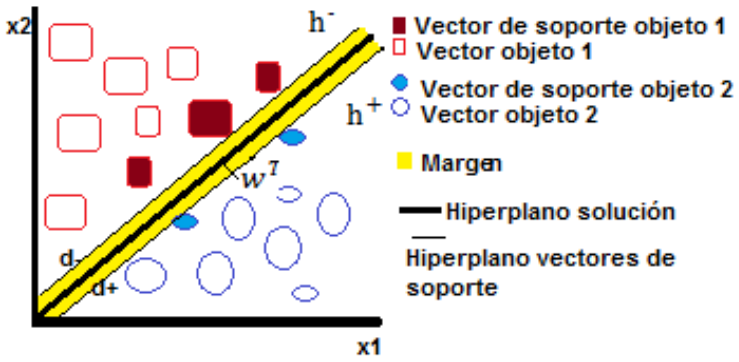


Figura 3.31: Representación Vectores de soporte

De esta forma, si esos vectores de soporte cambian, se desplazan o desaparecen, el resultado del SVM se ve modificado con ello. Así, si por ejemplo se decidiese desplazar estos vectores, el margen obtenido será mayor o menor en función del desplazamiento realizado. En el caso de, por ejemplo, cambiar aquellas muestras que no son los vectores de soporte, el sistema no sufriría ninguna modificación, obteniendo el mismo hiperplano como solución. Estas son algunas de las propiedades de los SVM, las cuales proporcionan una mayor robustez al clasificador, y evita el problema de overfiting o sobreentrenamiento, el cual se origina cuando el clasificador se centra demasiado en las muestras de entrenamiento, y no es capaz de establecer una función general para la división de conjuntos. Se podría entender como que en vez de aprender, memoriza una serie de características. En el caso del SVM, esto no es un problema porque se centra en esos vectores de soporte, como hemos podido comprobar.

Matemáticamente, que el SVM sea un clasificador lineal, significa que la solución es una expresión que puede corresponderse a la siguiente, según el número de dimensiones que estemos teniendo en cuenta:

$$w^T x_i + b = 0 \tag{3.16}$$

Siendo w un vector normal al hiperplano solución y el punto de intersección con el eje y, b, como podemos observar en la imagen anterior. Esta solución es obtenida a partir de los planos que contienen los vectores soporte, los cuales son denominados hiperplanos medios. Para que una muestra sea clasificada como positiva o negativa tendrá que cumplir lo que es denominado condición de clasificación, la cual implica que para las muestras positivas se tome el valor +1, mientras que para las muestras negativas esto sea -1. Poniendo en conjunto estas ideas podemos representarlo de la siguiente forma:

$$h^+ \rightarrow w^T x_i + b = +1(para\ el\ hiperplano\ positivo) \tag{3.17}$$

$$h^- \rightarrow w^T x_i + b = -1(para\ el\ hiper\ plano\ negativo) \tag{3.18}$$

Lo que combinándolo nos lleva a la siguiente condición de clasificación, que nos indica que cualquier muestra que siga esta regla estará correctamente clasificada:

$$\boldsymbol{y_i(w^T x_i + \mathbf{b}) \geq 1} \tag{3.19}$$

Una vez conocemos la información de esos planos medios, podemos calcular el tamaño total del margen, a partir de la distancia de los hiperplanos medios al hiperplano solución. De esta forma, siguiendo las reglas matemáticas de distancia entre planos obtenemos:

$$d^- = d^+ = \frac{|wx+b|}{\|w\|} = \frac{1}{\|w\|} \tag{3.20}$$

$$margen = d^- = d^+ = \frac{2}{\|w\|} \tag{3.21}$$

Puesto que la solución es la maximización del margen, esto se puede conseguir aplicando directamente los conceptos de la maximización. Sin embargo, con el fin de obtener relaciones matemáticas más sencillas, se puede invertir la relación y obtener la minimización de esta. Así, obtenemos el siguiente resultado:

$$\boldsymbol{Minimización\ \sigma(w) = \frac{1}{2}\|w\|^2} \tag{3.22}$$

$$\boldsymbol{siendo \|w\|^2 = w^T w} \tag{3.23}$$

A partir de esas dos ecuaciones podemos observar como el problema ha pasado a ser una simple optimización cuadrática, y que con las reglas matemáticas es bastante sencillo de resolver. Esta resolución suele ser la utilización de la regla de LaGrange, la cual proporciona una función auxiliar basada en la suma de la función a optimizar, lo que se corresponde en nuestro caso con la función $\sigma(w)$, más la suma ponderada o proporcional de las restricciones o condiciones de nuestro problema, lo que recogería la condición de clasificación. Esta proporcionalidad viene controlada por los multiplicadores de LaGrange, α. Así la solución al sistema planteado, o también llamado SVM Primal, tomaría la forma siguiente:

$$\max_{\alpha} \min_{w,b} \big(L(w,b,\alpha)\big) \begin{cases} \boldsymbol{Minimización\ \sigma(w) = \frac{1}{2}\|w\|^2} \\ \boldsymbol{y_i(w^T x_i + \mathbf{b}) \geq 1} \end{cases} \tag{3.24}$$

Así, desarrollando el Lagrangiano, minimizando esa nueva expresión con las derivadas parciales correspondientes, y sustituyendo en la expresión el valor de $w = \sum_i \alpha_i y_i x_i$, el cual es el valor obtenido a partir de las derivadas parciales, llegaríamos a los que se denomina SVM Dual, que es el nuevo sistema resultante de esas operaciones matemáticas:

$$\begin{cases} \boldsymbol{Maximización\ \partial(\alpha) = \frac{-1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_i \alpha_i} \\ \boldsymbol{\sum_i \alpha_i y_i = 0\,;\, \alpha_i \geq 0} \end{cases} \tag{3.25}$$

De donde sustituyendo obtenemos el clasificador final SVM:

$$\boldsymbol{f(x) = sgn(\sum_i \alpha_i y_i x^T x_i + b)} \tag{3.26}$$

Esta solución es la que nos da la información del hiperplano de separación en función de los vectores soporte y sus pesos respectivamente. Del valor de w, podemos comprobar como esta depende de los valores de las muestras, pero solo de aquellas que tengan el multiplicador de LaGrange mayor que cero. Esto es lo regulará que el resultado final sólo dependa de los vectores de soporte, y no de todos los del conjunto, puesto que estos tendrán una α nula.

Este sistema SVM dual consigue unos resultados prácticamente perfectos para casos linealmente separables, y además no implica la necesidad de cambiar parámetros puesto que la solución no es paramétrica. Sin embargo, en casos reales, los conjuntos no son tan fácilmente separables, y en ocasiones nos encontramos con solapamientos, al menos en dos dimensiones, de distintas piezas, lo que en un primer momento para el SVM puede ser una gran tarea. Lejos de que esto sea así, existen algunas técnicas que nos permiten solventar este tipo de dificultades gracias, entre otros factores, a que en el sistema SVM dual, las muestras están dispuestas como productos escalares por lo cual es más fácil su control o la modificación de ciertos parámetros, lo cual nos ayuda a introducir los siguientes conceptos:

- Kernel trick: el cual se basa en el mapeo de las distintas características a un espacio de dimensión superior, de forma que puedan ser distribuidas de una forma distinta, llegando incluso a eliminar esos solapamientos y consiguiendo una fácil división de los conjuntos, como se observa en la siguiente figura:

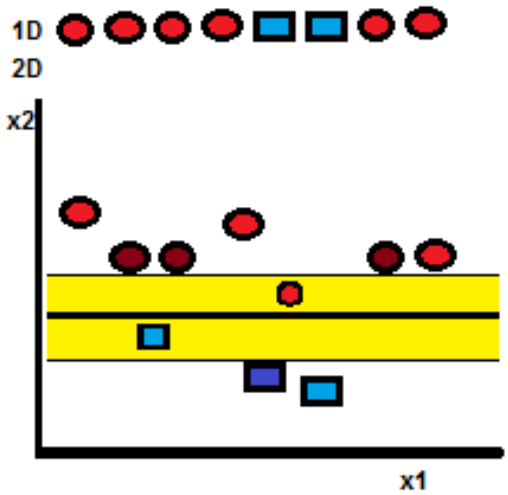


Figura 3.32: Truco Kernel

Matemáticamente, gracias a que en la SVM dual nos encontramos las muestras por productos escalares, nos va a resultar más sencillo introducir esta función Kernel correspondiente al mapeo realizado. Así este sería de la forma:

$$x \rightarrow \varphi(x)$$

$$\boldsymbol{K(x,z) = \varphi(x)^T \varphi(z)} \tag{3.27}$$

E integrándolo en la SVM dual, quedaría el siguiente resultado:

$$\begin{cases} \boldsymbol{Maximización\ \partial(\alpha) = \frac{-1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathrm{K}(x_i, x_j) + \sum_i \alpha_i} \\ \boldsymbol{\sum_i \alpha_i y_i = 0\,;\, \alpha_i \geq 0} \\ \boldsymbol{f(x) = sgn(\sum_i \alpha_i y_i \mathrm{K}(x_i, x_j) + b)} \end{cases} \tag{3.28}$$

donde el término intependiente:

$$\boldsymbol{b = y_i - w^T \varphi(x_I) = y_i - \sum \alpha_I y_j \mathrm{K}(x_i, x_j)} \tag{3.29}$$

puede obtenerse al igualar $f(x)$ a cero

Ya existen algunos Kernels diseñados, los cuales hacen el trabajo más fácil. Así por ejemplo, podemos contar con:

$$\boldsymbol{Polinomial: K(x,z) = \langle x, z \rangle^d} \tag{3.30}$$

$$\boldsymbol{Radial: K(x,z) = e^{-\|x-z\|^2/2\sigma}} \tag{3.31}$$

$$\boldsymbol{Interseccion: K(x,z) = \sum_{i=1}^{n} min(x(i), z(i))} \tag{3.32}$$

Sin embargo tememos que tener en cuenta que el uso de estos Kernels supone la introducción de nuevos parámetros al entrenamiento que debemos de controlar, como es el caso del parámetro d, del Kernel polinomio, el cual nos indica cual es el grado del polinomio.

- Margen suave: que es una condición que permite la existencia de muestras de ambos conjuntos en el espacio designado como solución, de ahí su nombre de suavizado del margen. De esta forma pueden aparecer más falsos positivos o negativos, pero se puede llegar a conseguir algunas detecciones aceptables. Esto en la mayoría de las ocasiones puede dar mayor robustez y permite más tolerancia a los errores.

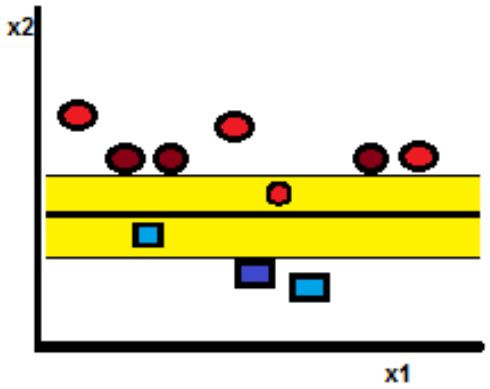


Figura 3.33: Margen Suave

Esas muestras que se encontraran dentro del margen designado serán llamadas variables de holgura, y serán elementos de los que dependerá nuestra solución. Estarán designadas por δ_i, siendo i el número de muestras que violan la condición del margen, en el caso del a figura 3.33, 2. Su incorporación en el sistema SVM quedará definida por la modificación de la condición de clasificación y vendrá de la mano de un factor C que regulará su relevancia en el sistema, como se puede ver en el siguiente sistema, el cual ya cuenta con la posibilidad de cambio de Kernel:

$$\begin{cases} Maximización\ \partial(\alpha) = \frac{-1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathrm{K}(x_i, x_j) + \sum_i \alpha_i \\ \sum_i \alpha_i y_i = 0\,; \alpha_i \geq 0 \\ 0 \leq \alpha_i \leq C \quad b = y_i(1-\delta_i) - \sum \alpha_I y_j \mathrm{K}(x_i, x_j) \\ \boldsymbol{f(x) = sgn\left(\sum_i \alpha_i y_i \mathrm{K}(x_i, x_j) + b\right)} \end{cases} \tag{3.33}$$

Como se puede ver, en la solución final del SVM, no tiene un gran impacto directo. Sin embargo, para un perfecto funcionamiento este se tendrá que controlar de forma que se encuentre el equilibrio entre maximizar el margen y minimizar el error, al hacer nuestro clasificador menos preciso.

3.2.2 Método de Viola Jones

En este caso, este método creado por los investigadores Paul Viola y Michael Jones nació en 2001, algunos años antes que el de Dalal y Triggs [3]. Es un sistema que a día de hoy ha sido utilizado para infinidad de detecciones de objetos, pero que en su día se especializó en la detección de caras. Una de sus características principales, lo que hace que sea un método que se siga utilizando a día de hoy ampliamente, es su robustez y su capacidad de trabajar a tiempo real. Es por ello, que por ejemplo, este sea el procedimiento utilizado en algunas aplicaciones de móviles con detección de caras incorporadas.

Este, al igual que el método de Dalal y Triggs [3], y casi todos los sistemas basados en aprendizaje supervisado, vuelve a trabajar con un vector descriptor y una máquina que permite la clasificación de los objetos. Sin embargo, este cuenta con unas herramientas peculiares que se complementan perfectamente entre sí, y que hacen que el sistema de Viola Jones [2], como es popularmente conocido, sea realmente especial. Fue un sistema con una cascada de clasificadores de 38 niveles, que trabajaba con 6060 características y ofrecía resultados de falsos positivos menores al 0,001% y más de un 95% de detecciones correctas.

La generación del vector descriptor es totalmente distinta al procedimiento seguido por HOG. En este caso, las características se obtendrán a partir de los métodos de Haar. Estos son un conjunto de filtros que son recorridos por las imágenes, en diferentes escalas, que a partir de unas sencillas relaciones matemáticas, extraen la información de las imágenes. Para una misma imagen se recorren un gran número de filtros, lo que hace que el número de características sea muy elevado, del orden de más de 100 mil.

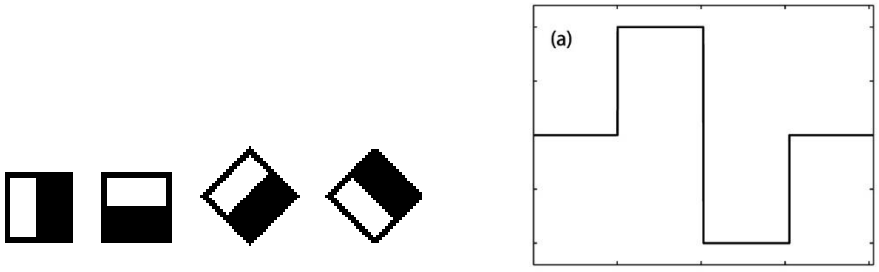
En el paso de clasificación, el número de características a tratar será extremadamente superior a lo que otros métodos manejan, y por eso la selección del Adaboost. Este está especializado, no solo en el aprendizaje del sistema, sino además en la selección de las características, de forma que, conforme se está entrenando, el sistema va desechando aquellas propiedades que no son relevantes. Además, como se explicará a continuación, la solución final del clasificador se conseguirá a través de varios procesos, por lo que aquellos en última posición tratarán solo con las características y muestras que no han sido eliminadas.

Vamos a realizar una explicación más clara de estos dos procesos, así como de la obtención final del clasificador, para saber con certeza cuales son las diferencias entre estos dos métodos que estamos estudiando en el trabajo.

3.2.2.1 Descriptor Haar

Para obtener el descriptor de Haar es necesaria la aplicación de una serie de filtros, a distintas escalas, a todas y cada una de las imágenes de muestra o de entrenamiento con la que estemos trabajando. Estos filtros, conocidos popularmente como los filtros de Haar, nos permitirán obtener las características relevantes de las muestras, como pueden ser los contornos verticales, horizontales o diagonales, tal y como perseguíamos con los gradientes de HOG.

Estos filtros obtienen su nombre a partir del gran parecido entre ellos y los wavelets de Haar, como se observa en la figura 3.34, los cuales son una cierta secuencia de funciones que produce representaciones entre 0 y 1 en la escala real.



(a) (b)

Figura 3.34: (a) Filtros Haar [15] (b) Wavelets [16]

Como se puede observar en la imagen superior, estos filtros están compuestos por unas estructuras formadas por cuadrados blancos y negros, que se disponen adyacentemente unos de os otros tanto horizontalmente como verticalmente. Podemos diferenciar los filtros básicos, que son los de la imagen de arriba, los cuales cuentan con un número de cuadrados entre 2 y 4, y los filtros extendidos, que son estructuras un poco más complejas y con distintas orientaciones. Los cuadrados pintados de negro serna considerados los positivos del filtro y los blancos los negativos. De esta forma, al aplicar el filtro, el resultado de este será la suma de contribuciones de intensidades de los pixeles de cada color, teniendo en cuenta que los blancos van marcados por un signo negativo. La solución de aplicar estos filtros es la información de los distintos cambios de intensidad en la imagen. Al utilizar tantos filtros en numerosos tamaños, lo que se intenta es estudiar todas las posibilidades de cambio, desde horizontal o vertical, hasta diagonal. La relación matemática seria la siguiente.

$$\boldsymbol{I(x,y) = \sum_{(x,y)\in N} I(x,y) - \sum_{(x,y)\in B} I(x,y)} \qquad (3.34)$$

siendo N y B los conjuntos de pixeles negros y blancos respectivamente

Dependiendo del tipo de filtro que se utilice o del tamaño de este, se obtendrán imágenes donde se resalten unas características u otras, así por ejemplo con un filtro de dos cuadrados, uno de cada color, dispuestos horizontalmente, se detectaran los bordes verticales, mientras que si ese mismo filtro se reescala obteniendo uno de dos cuadrados más grandes, los resultados serán detecciones de zonas verticales a escalas mayores. Aplicando este último a los coches, se obtendrían por ejemplo, los bordes verticales de los cristales.

Como ya se ha comentado, todos los filtros serán aplicados a todas las imágenes en todos los tamaños que esta permita. Esto viene condicionado por la cantidad de partes negras y blancas del filtro, ya que deber ser equivalentes. En aquellas imágenes que un tamaño más del filtro no permita esta condición, simplemente se pasará a una nueva estructura. El resultado de aplicar este procedimiento es lo que genera las características Haar que serán utilizadas para entrenar el clasificador. Sin embargo, no nos podemos

olvidar que el número de características obtenidas de esta forma es muy elevado, por lo que tendremos que realizar algunos cambios para poder trabajar con ellas tranquilamente. Para ello recurrimos al concepto de imagen integral, con la cual, trabajar con un número elevado de características deja de ser un problema.

La imagen integral puede definirse como una nueva imagen, fruto de una transformación mediante la cual el valor de un pixel es equivalente a la suma de todos los pixeles en la imagen original que están situados a su izquierda y encima. Cabe recordar que el centro de coordenadas en una imagen se encuentra en la esquina superior izquierda, encontrándose en el eje horizontal el correspondiente al "x" y en el eje vertical el del "y". Esto puede representarse mediante las siguientes expresiones:

$$\boldsymbol{II(x,y) = II(x,y-1) + s(x,y)} \tag{3.35}$$

$$\boldsymbol{s(x,y) = \sum_{x' \leq x} I(x',y) = s(x-1,y) + I(x,y)} \tag{3.36}$$

$$\begin{gathered} donde: \\ II(x,y): el\ valor\ del\ pixel\ actual \\ II(x,y-1): el\ valor\ del\ pixel\ inmediatamente \\ encima\ del\ que\ estudiamos \\ s(x,y): el\ valor\ de\ la\ final\ del\ pixel\ a\ estudiar \end{gathered}$$

Como podemos observar en la imagen 3.35, para calcular el valor del pixel amarillo, es necesario sumar la del pixel encima de él (el rojo), que equivale a toda la región azul, más la fila de pixeles a la izquierda del amarillo (la zona verde).

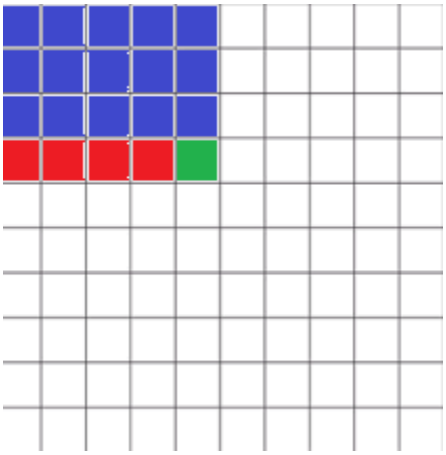


Figura 3.35: Zonas imagen integral

Este método es muy efectivo, puesto que como ya hemos visto anteriormente, al recorrer la imagen con los filtros, se realiza la suma de contribuciones de cada color. De esta forma, teniendo los valores de los pixeles anteriores, los cálculos pueden hacerse de

forma mucho más rápida, gracias a una aproximación geométrica. Dada la imagen siguiente, supongamos que queremos calcular la zona A. Conociendo el valor de los pixeles P1, P2, P3 y P4, con unas simples operaciones se puede obtener.

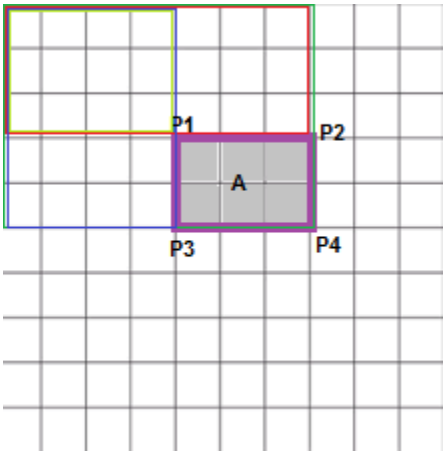


Figura 3.36: Cálculo imagen integral

Esta es la operación resultante y con la que obtendremos el resultado de la zona A.

$$\boldsymbol{A = II(P_4) - II(P_2) - II(P_3) + II(P_1)} \tag{3.37}$$

Esto también puede aplicarse a los filtros extendidos, pero de una forma distinta. Estos filtros, que en su mayoría tienen una orientación de 45º, necesitarían un procedimiento similar, y diferencia de la orientación de los puntos y las zonas de los que estos toman los valores. Al desarrollar las relaciones geométricas, obtendremos el mismo resultado que la figura 3.36, pero con designación de puntos distintas, como se puede observar en las imágenes 3.37 y 3.38.

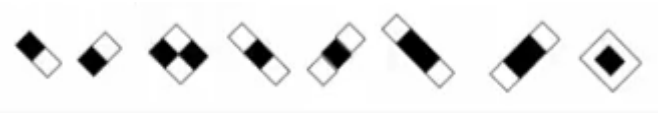


Figura 3.37: Filtros extendidos [14]

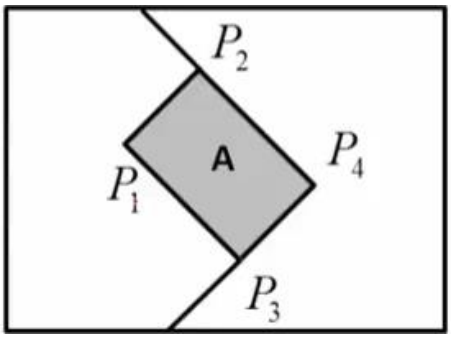


Figura 3.38: Aplicación filtros extendidos [14]

3.2.2.2 Adaboost

Adaboost, o adaptative boosting, es un método de clasificación, distinto al SVM, que nos permite realizar las calificaciones trabajando con un número elevado de características. Es un método que para encontrar la solución no se basa en la búsqueda de esa única función o línea que separa los dos conjuntos como ocurría en el SVM, si no que consiste en el sumatorio de los resultados de numerosos clasificadores simples, con lo cual se consigue la solución, un gran clasificador muy robusto, que utiliza solo las características más relevantes de estas.

Como se ha mencionado, Adaboost, no trata de buscar la función que permita separar linealmente los conjuntos, sino que es un clasificador resultado de la combinación de pequeñas comprobaciones y comparaciones de las características. Esto hace, que para conjuntos linealmente no separables, se encuentren soluciones muy precisas sin necesidad de recurrir al mapeo del espacio o a la suavización de la condición de clasificación. Esta es una de sus principales fundamentos. Los clasificadores simples, por separado no ofrecen gran información, pero la combinación de estos, y la otorgación proporcionada de una serie de pesos a las muestras mal clasificadas, hace que el resultado final adquiera la robustez esperada.

A escalas generales, Adaboost consigue una característica para cada filtro en cada tamaño. Así, una vez realizada esta operación, se dividen las muestras en dos conjuntos, positivos y negativos, según un umbral establecido, o lo que es conocido como frontera de decisión. Aquellas piezas que no hayan sido correctamente clasificadas, pasarán a tener un peso mayor, de cara a una nueva aplicación de un filtro, mientras que las que sí lo han sido, ven reducida su prioridad. Acto seguido, se vuelve a aplicar un filtro, que nos dará una nueva frontera, y se realiza el mismo procedimiento. Al final del proceso, aquellas zonas que se hayan clasificado un número mayor de veces como positivas o negativas, quedarán finalmente bajo esa etiqueta, es decir, el resultado final será la suma de las distintas contribuciones de los clasificadores simples aplicados a la imagen. Esto se verá explicado más explícitamente a continuación.

Como se puede deducirse, este método se centra en el trabajo que realiza, en los primeros procesos, lo que se conoce como clasificadores simples. Un clasificador simple es lo que se denomina un árbol de decisión binaria de profundidad 1. Este árbol persigue buscar una función para cada decisión a realizar, y establecer un umbral que nos permitirá apropiarle un valor u otro a las muestras con las que trabajemos. Cada clasificador simple tendrá su fundamento en una única característica Haar, lo que proviene de aplicar un único filtro en una única escala. De esta forma, esto puede escribirse como la relación siguiente:

$$\boldsymbol{h(x)} = \begin{cases} -\boldsymbol{\alpha}, \wedge \boldsymbol{f(x)} < \boldsymbol{\theta} \\ \boldsymbol{\alpha}, \wedge \boldsymbol{f(x)} \geq \boldsymbol{\theta} \end{cases} \tag{3.38}$$

$$donde:$$
$$\alpha = \{-1,1\}\ valor\ de\ la\ clasificación$$
$$f(x): función\ de\ características$$
$$\theta: umbral$$

Para establecer tanto el umbral, como la designación del signo del parámetro α, ya que este solo toma el valor de la unidad, será necesario estudiar atentamente los valores obtenidos para las muestras, y sus respectivos pesos. Así, se intentará buscar un umbral que minimice el error en la obtención de conjuntos, intentando clasificar correctamente todas las muestras posibles. Para ello entrará en juego la correcta asignación de pesos, según las muestras se hayan etiquetado adecuadamente o no. Para ello un método es ordenar las muestras según el valor de la característica recién calculada y el peso proporcional. De esta forma se pueden calcular todos los umbrales posibles, junto con el error, y averiguar cuál es la mejor opción.

Sin embargo, esto sólo nos permitiría entrenar un clasificador simple de una característica. Como ya se ha mencionado, una imagen contaría con muchas características y por lo tanto con muchos clasificadores simples, pero de todos ellos tenemos que elegir las más relevantes, las que pasarán a un clasificador mayor. Para ello nos vamos a decantar por aquellas características que, una vez obtenidas el umbral adecuado, presentan el menor error de clasificación. Es decir, en cada iteración se entrenará un clasificador simple para cada característica, y al final del proceso, de cada iteración solo obtendremos una característica relevante, la que tenga menos error de clasificación. Aquí queda recogida la capacidad de entrenar y seleccionar que se comentaba el principio de este método, puesto que además de entrenar unas nuevas funciones de características, estamos eligiendo la mejor opción de todas ellas, el mejor clasificador simple.

Esta solución obtenida es la que se incluirá como parte del clasificador final. Sin embargo, cada nueva incorporación no tiene la misma importancia que los clasificadores simples obtenidos en previas iteraciones, ya que, entre otros factores, los errores de clasificación van a ser distintos. Así, en cada iteración, cuando se aprenda un nuevo clasificador, el peso iterativo de cada una de las anteriores será modificado según el valor del peso anterior y de un parámetro de clasificación que toma el valor del error del último aprendido. Así, la variable de actualización del peso puede quedar recogida de la siguiente manera:

$$\boldsymbol{w_i(t+1)} \begin{cases} \frac{\boldsymbol{w_i(t)}}{\boldsymbol{2\varepsilon_t}} & \boldsymbol{h(x_i) \neq y_i (mal\ clasificados)} \\ \frac{\boldsymbol{w_i(t)}}{\boldsymbol{2(1-\varepsilon_t)}} & \boldsymbol{h(x_i) = y_i (bien\ clasificadas)} \end{cases} \tag{3.39}$$

$$siendo\ \varepsilon_t\ el\ error\ del\ nuevo$$

De esta forma, el peso de aquellos ejemplos del nuevo clasificador que han quedado mal clasificados, se verá aumentado, frente al de aquellos que si han sido bien clasificados, el cual se reducirá según la variable de actualización que acabamos de explicar. La incorporación del factor $\frac{1}{2}$ proviene de suponer que el clasificador obtenido será mejor que uno aleatorio, el cual tendría un $\epsilon_t = 0.5$

Con el fin de conseguir el clasificador fuerte, es necesario establecer un número de iteraciones, por lo que así además obtendremos un número concreto de características relevantes, y para obtener la solución final será necesario computar la suma de todas esas funciones de características obtenidas ponderadas con una relación basada en sus correspondientes errores de clasificación, otorgándole mayor o menor importancia según su eficacia al clasificar. El resultado será el signo que se le designará a la clasificación, siendo positivo si es mayor que cero o negativo si tiene un valor menor.

$$\boldsymbol{H(x) = sign\left(\sum_{i=1}^{T} \alpha_i h_i\,(x)\right)} \tag{3.40}$$

$$\boldsymbol{donde: \alpha_t = \frac{1}{2} log \frac{1-\epsilon_t}{\epsilon_t}\ y\ T\ es\ el\ número\ de\ características}$$

Esto, denominado clasificador fuerte, es lo que se utilizará como parte de la cascada general de clasificadores, la cual mediante el entrenamiento adecuado, nos proporcionara la respuesta final de objeto o no objeto.

Una cascada de clasificadores es un método de entrenamiento mediante el cual una muestra es clasificada como objeto, si y solo si, todos los clasificadores que la componen la consideran objeto. Es decir, la imagen seria recorrida con cada clasificador, o lo que se denomina nivel, y etiquetada por ellos. Si en uno de ellos la muestra es desechada, está directamente es eliminada de las posibles soluciones para "objeto sí". Esto es lo que hace que el método de Adaboost sea tan robusto y rápido. El esquema de la cascada de diagramas puede observarse en la figura 3.39. Como se ve, en los clasificadores iniciales, los cuales sólo tienen en cuenta algunas características, son eliminadas gran parte de las muestras que no son objeto. Así, clasificadores superiores trabajarán con menos imágenes, pero a su vez con más características, lo que permite un cómputo más rápido al número de muestras ir disminuyendo, y siendo cada vez más preciso.

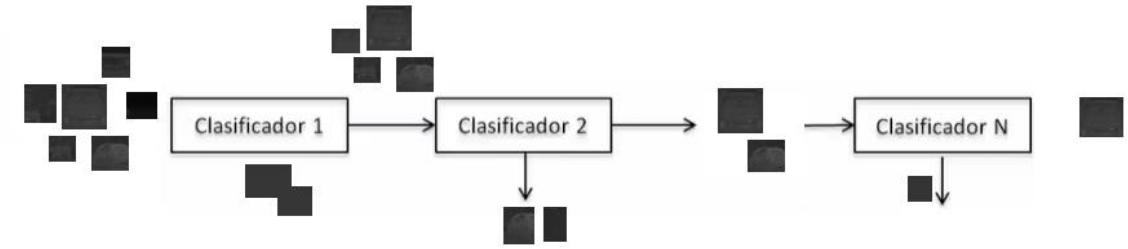


Figura 3.39: Cascada de clasificadores

La cascada de clasificadores, además de los parámetros que pueda contener cada clasificador fuerte, cuenta con un par de parámetros característicos a tener en cuenta en cada nivel. Estos son los que permitirán el control de los niveles, y por extrapolación, del sistema de cascadas. Estos son el número máximo de falsas detecciones que permitiremos y el número mínimo de detecciones correctas que exigiremos para cada clasificador. De esta forma, conforme vayamos avanzando con una muestra por los niveles, estos, en proporción, nos indicarán la dureza o la suavidad total del clasificador.

Por ejemplo, si permitimos un 50% de detecciones falsas y un mínimo del 99% de detecciones verdaderas, en el segundo nivel, estos podrían corresponder a un 25% de detecciones falsas detectadas y a un 95% de mínimas detectadas, aunque los valores reales de cada clasificador sean los mismos. Esto entra en conflicto con la explicación de la generación de un clasificador fuerte, puesto que en aquel caso el objetivo global era conseguir el menor error de clasificación posible, y lo que se busca ahora, como entrenamiento de la cascada, es que cada clasificador sea capaz de cumplir con las detecciones mínimas positivas y las máximas negativas que se estipulan. Es por ello, que se modificarán ciertos aspectos para conseguir que esto pueda cumplirse.

De esta forma, para cada clasificador fuerte se establecerán unas normas o permisiones que ayudarán a optimizar esos dos parámetros que estamos buscando. Así introducimos el concepto de margen, el cual nos permitirá modificar el umbral que se estipuló en la generación del clasificador fuerte, con el fin de permitir esas detecciones máximas y mínimas. Para ello se incluye un nuevo parámetro s, que puede considerarse un factor de margen, que nos controlará cuando ha de moverse la frontera de decisión y hacia qué dirección. Este parámetro tomará su valor gracias a un proceso iterativo.

$$\boldsymbol{H(x) = sign\left(\sum_{i=1}^{T} \alpha_i h_i\,(x) + s\right)} \tag{3.41}$$

Al mover la frontera mediante el controlador s, es posible que en ocasiones no encuentre el valor óptimo para cumplir con ambas condiciones de detección. Cuando la frontera se desplaza en busca de mayores detecciones positivas, puede ser que deje paso a mayor número de detecciones negativas y viceversa. En este caso, ese clasificador se desecharía y se generaría un nuevo clasificador fuerte con un número mayor de

características, es decir, de clasificadores simples, mediante las cuales podemos hacerlo más fuerte, y por lo tanto conseguir que cumpla con los objetivos.

$$\boldsymbol{H(x) = sign\left(\sum_{i=1}^{T'} \alpha_i h_i\,(x) + s\right)\ donde\ T' > T} \tag{3.42}$$

Incluso aumentando el número de características, es posible que no se encuentre solución posible para resolver el problema. Para evitar que el sistema se quede constantemente buscando una solución gracias al aumento de características, se establece un número máximo de estas para todos los clasificadores.

Así, el funcionamiento general de la cascada, a niveles de aprendizaje, se podría resumir en:

1. Establecimiento de un número máximo de detecciones negativas y uno mínimo de detecciones positivas.

2. Generación de un clasificador fuerte.

3. Búsqueda de la optimización de esos parámetros. Si no se consiguen los objetivos, añadir características al clasificador fuerte hasta cumplir con las condiciones establecidas o alcanzar el límite de características de un clasificador.

4. Si se completa el paso tres, estaríamos ante el clasificador fuerte número 1. El procedimiento se realizaría de igual forma para seguir añadiendo clasificadores. Sin embargo, para mejorar la precisión, aquellas imágenes que han sido detectadas como no-objeto en el clasificador 1, será introducidas como imágenes negativas en el clasificador 2 y así sucesivamente.

5. El entrenamiento finaliza cuando hayamos alcanzado el número máximo de niveles establecidos o cuando los requisitos de detecciones máximas y mínimas hayan sido cumplidos.

4 Desarrollo del trabajo

Una vez entendida la aplicación de nuestro sistema a diseñar, así como, una vez entendido el procedimiento que existe detrás de cada método a aplicar, podemos pasar a la generación de esos detectores, con el fin de, además de encontrar el mejor detector de los dos métodos, así como las configuraciones que se puedan ajustar a nuestra aplicación, de poder entender y emplear todo lo que anteriormente se ha comentado. Se va a estudiar cada parámetro a fondo con el objetivo de conocer el impacto que estos pueden ocasionar en los sistemas de detección. Se intentará encontrar el equilibrio entre verdaderos positivos y falsos negativos, así como los falsos negativos que puedan existir.

Para ambos métodos se ha hecho uso de las librerías OpenCV, las cuales han proporcionado gran información y métodos para la realización de este proyecto. Esta es una librería de código abierto, la cual está creada por los mismos usuarios, ya que se pueden incluir funciones que supongan un gran avance para el mundo de la visión por computador. Este fue el caso de los métodos a utilizar, lo cuales ya tienen funciones representativas en esta librería. Sin embargo, como se comentará posteriormente, se han tenido que realizar algunos cambios en los entrenamientos, y hacer uso de otras librerías, SVMLight, en el caso del método de Dalal y Triggs [3], por problemas de incompatibilidades con los resultados.

Ambos proyectos, por lo general, se han realizado en C++ gracias al entorno de trabajo de QtCreator. Se ha utilizado el sistema operativo de Ubuntu 12.04, ya que existía mayor compatibilidad con los códigos de programación utilizados. En el anexo I de esta memoria se encuentran los detalles para la instalación de las librerías OpenCV en Ubuntu junto con el programa QtCreator. En el caso del método de Viola Jones [2], para el entrenamiento no ha sido necesario utilizar este programa, ya que podía ser ejecutado desde la Terminal, pero para la configuración final del detector, donde se han realizado las diferentes pruebas para el análisis, sí se ha recurrido a él.

En un principio se comentará el trabajo previo a los entrenamientos, que, principalmente, consistió en el procesamiento de las imágenes que nos servirían como conjunto de entrenamiento y, posteriormente, se detallarán por separado el trabajo realizado para cada método, así como los parámetros utilizados y estudiados, al igual que se mostrarán, en cada caso, los resultados de las detecciones. De cada método se detallará el proceso completo hasta la obtención de los detectores, así como las diferencias entre cada uno de ellos o los problemas encontrados durante el camino.

4.1 Trabajo previo

Antes de realizar cualquier entrenamiento, es necesario obtener las imágenes de prueba, o conjunto de entrenamiento, sobre las que íbamos a fundamentar nuestros detectores. Para ello fue necesario tomar las imágenes con la cámara del coche de la universidad.

Para esto era necesario conectar el sistema que lleva incorporado el vehículo, de forma que las secuencias que fuese captando por la cámara, se almacenasen en un disco duro. Fueron necesarios varios días para la toma de las imágenes necesarias, ya que en alguna ocasión se encontraron problemas como la mala calibración de la cámara o la pobre calidad de las imágenes debido a las condiciones atmosféricas. Como se ha comentado, las cámaras infrarrojas se ven afectadas por las altas temperaturas, puesto que se complica notablemente la distorsión de la realidad captada. Se consiguió un total de 10.000 imágenes, las cuales fueron tomadas con una diferencia de unos 100 ms y representaban todo lo que la cámara tenía a su alcance. Algunos ejemplos serían las siguientes secuencias de la figura 4.1.

Figura 4.1: Secuencias tomadas

Así, podemos observar como en cada imagen se puede encontrar distintos vehículos, desde turismos a camiones o autobuses, hasta imágenes donde sólo exista carretera o peatones. Sin embargo, ambos modelos cuentan con la misma entrada de información, dos tipos de imágenes bien diferenciadas, las imágenes positivas, aquellas que únicamente recogen la información del objeto, lo que en este caso se correspondía a vehículos, y las imágenes negativas, las cuales pueden representar cualquier elemento a excepción de vehículos. Para ello, fue necesario el preprocesado de las imágenes para conseguir, de lo obtenido por la cámara infrarroja, las muestras que se adecuasen a nuestras necesidades.

4.1.1 Imágenes Positivas

Para obtener las imágenes positivas tal y como eran pedidas para nuestros métodos, según los códigos que se utilizaron como se explicará más adelante, fue necesario seguir los siguientes pasos:

- Etiquetado

  Era necesario realizar los recortes de vehículos de esas secuencias que se obtuvieron. Para ello, fue necesario desplazarse manualmente en cada imagen de las mencionadas y remarcar cada coche. El código utilizado, permitía guardar la información de la situación del vehículo en una base de datos SQL, siendo esta el punto de la esquina superior izquierda, el ancho y el alto. Esta base de datos fue utilizada para las imágenes de entrada del HOG en el paso siguiente, con el fin de reestablecer los márgenes de etiquetado, ya que era conveniente que todas las imágenes contasen con unas características similares. Así, de esas 10.000 imágenes, se pudieron obtener entorno a unas 800 imágenes de positivas, o de coches, que se convertirían en el doble, ya que como entrada de positivas utilizábamos las originales y sus correspondientes al espejo con respecto al eje y, tal y como se muestra en la figura 4.2.

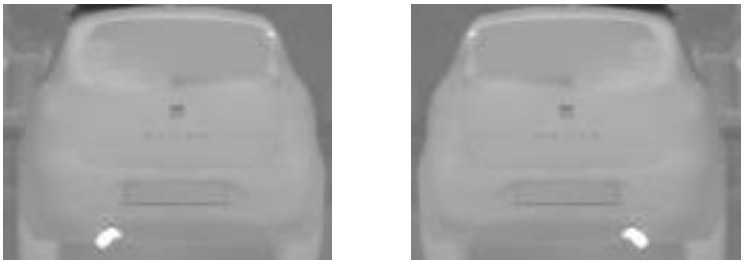


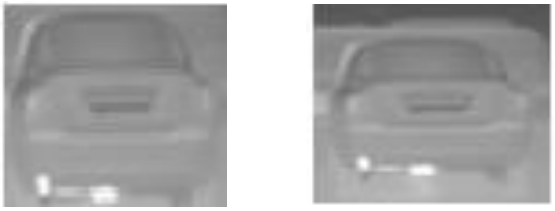
Figura 4.2: Ejemplo de imágenes espejo

- Establecimiento de márgenes

  Tras esto fue necesario establecer los bordes o márgenes de interés a dejar en las imágenes. Esto se usó para el caso del HOG, ya que para calcular un gradiente, debe existir un cambio de intensidad, y para que esto sea posible, es necesario que exista un margen que permita diferenciar perfectamente el borde del objeto y lo que ya no forma parte de este, para que así. Como se ha mencionado, se recurrió al uso de la base de datos para acceder a la posición de cada coche en cada imagen inicial. Los vehículos a detectar serían aquellos que desde nuevo vehículo pudiésemos ver delante de nosotros, tanto si estos circulaban en nuestro mismo sentido y por lo tanto veríamos la trasera del vehículo, como si iban en el sentido opuesto, y por lo tanto veríamos la delantera. Los coches que no se intentarían detectar serian aquellos que circulasen en una dirección perpendicular a la nuestra, aquellos de los que viésemos los laterales del automóvil.

  Partiendo de esa base, podemos observar por las imágenes anteriores, que tanto las traseras como las delanteras de los vehículos, tienen una forma cuadrada o rectangular. Con el fin de encontrar características similares a los vehículos, se estudió como el ancho del vehículo, para la mayoría de los modelos, suele ser el mismo, o entorno a los mismos valores, mientras que para el caso del alto, esto

no siempre se aplica, puesto que puede verse modificado según el modelo de coche con el que estemos tratando. Además de esto, teníamos que tener en cuenta que la anchura, por lo general, es mayor que la altura. Es por ello que el establecimiento de los márgenes se fijaría para uno de estos parámetros, en este caso el ancho, y se reescalaria de forma que no se obtuviesen deformaciones, y todas las imágenes saliesen con un tamaño concreto. En el caso de, por ejemplo, un tamaño de imagen de 64x64 píxeles, el margen horizontal se establecería alrededor de 16 píxeles, dejando 8 píxeles por cada lado, mientras que para el vertical se ajustaría a las exigencias del alto a 64 píxeles. Esto puede observarse en las figuras 4.3a y 4.3b, en las cuales se muestran los efectos de aplicar el reescalado directamente o mediante el método correcto, respectivamente.



(a) (b)

Figura 4.3: (a) Imagen distorsionada (b) Imagen con márgenes correctos

- Reescalado
  Las imágenes era reescaladas a los tamaños de entrenamiento necesarios, siendo estos 64x64 píxeles, 48x48 píxeles y 32x32 píxeles.

Para el caso del método de Viola Jones [2], en las imágenes positivas no era necesario establecer ese margen del objeto, sino que únicamente es necesario la figura del objeto. Es por ello que las imágenes introducidas en el segundo método, eran los recortes procedentes del etiquetado.

4.1.2 Imágenes Negativas

Para en caso de las imágenes negativas, ejemplo de las cuales se pueden ver en la figura 4.4, el proceso era mucho más simple que el anterior, solo eran necesarias imágenes o recortes de imágenes donde no apareciesen vehículos. Para este caso, se tomaron algunas imágenes, sin coches, y se realizó un proceso de desplazamiento de ventana para obtener pequeñas secciones de esta del tamaño deseado.

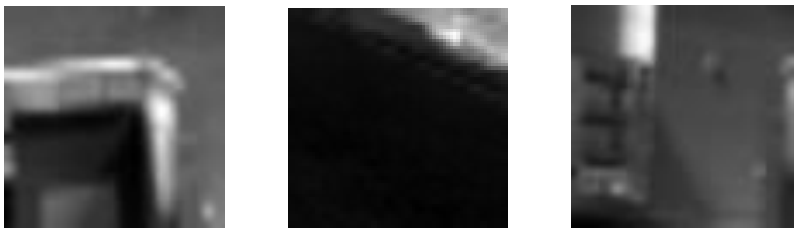


Figura 4.4: Ejemplos imágenes negativas

Además, posteriormente, se transformaron a tamaños de 32x32 píxeles, 48x48 píxeles y 64x64 píxeles, puesto que tanto las positivas como las negativas, debían estar al mismo tamaño. En el caso del método de Viola Jones [2], esto no era una condición restrictiva, pero si aconsejada. Tras esto, las imágenes estaban listas para ser utilizadas y poder así entrenar nuestros detectores.

4.2 Método 1: SVM junto a HOG

Como se ha mencionado, el primer método a estudiar fue la combinación de los descriptores de HOG y la máquina de vectores SVM. En un primer momento se pensó en la posibilidad de empezar un proyecto desde cero, desarrollando cada una de las funciones necesarias para realizar la extracción de características y los entrenamientos, sin embargo, a pesar de querer utilizar las libreras OpenCV desde un principio para desarrollar todo el algoritmo, nos encontramos con muchas dificultades en el camino.

Estas tienen incorporadas las clases HOG y SVM, pero cuentan con el problema de no ser compatibles entre sí. Al desarrollar el descriptor HOG y hacer uso de él para conseguir el clasificador, o como se ha explicado anteriormente, el hiperplano de separación, nos encontramos que las funciones propias de OpenCV de la clase SVM no ofrecían como resultado ese hiperplano, si no los vectores de soporte. Ante la dificultad en el trabajo, y tras encontrar numerosa ayuda al respecto, recurrimos a otro camino.

En el año 2012, frente al crecimiento de los sistemas de detección mediante este método, gracias al reciente articulo del Dalal y Triggs [3], y debido a las complicaciones de trabajar directamente con OpenCV, Jan Hendriks [17], un investigador del campo de visión artificial, desarrollo un código [17] que permitía, gracias a unas libreras externas, combinar los que Dalal y Triggs [3] exponían. De esta forma, gracias a unas simples indicaciones de introducción de las imágenes y del cambio de algunos parámetros, todo usuario con acceso al código podía desarrollar un clasificador sin encontrarse con los problemas anteriores.

Para el correcto funcionamiento de este proyecto, hacía falta la instalación de las librerías correspondientes al SVM. Para ello, desde el año 2008, existe una página especializada [18] en ese proyecto, la cual va avanzando con el paso de los años,

adaptando los recursos a las necesidades de cada momento. Así, esta cuenta con más de 3 tipos de SVM, entre lo que se pueden encontrar SVMRank, SVMStruct o SVMLight, la cual es la que se va a utilizar en este caso.

El fichero comprimido que se proporciona en esta página es un conjunto de archivos *sources*, *headers* y *objects*, los cuales serán incluidos en el proyecto de Jan Hendriks [17], trainHOGmaster. Este contaba con una carpeta exclusiva para la inclusión de estos, llamada *svmlight*, en la que ya se encontraba un previo svmlight.h. Sin embargo, hace falta realizar una serie de pasos mediante la Terminal para obtener todos los necesarios, entre los que se encuentran los archivos de tipo objeto. Para ello, situados en la carpeta correspondiente, ejecutamos el comando *make all*, para que estos se generen. Todos ellos, no son más que el conjunto de procedimientos que hacen posible encontrar el hiperplano de solución de nuestro conjunto. Además de incluirlos en la carpeta de nuestro proyecto, para que permitiesen su funcionamiento, fue necesario vincularlos al proyecto en cuestión incluyendo la dirección de los archivos del SVM en el .pro del proyecto.

Después de estos pasos, casi se cumplirían los requisitos necesarios para comenzar los entrenamientos. Cada modelo contaría con parámetros propios que difieren de la conducta de los demás. De esta forma, además de buscar la mejor solución, también estudiaríamos el comportamiento de cada uno de ellos. Estos cambios o pruebas van a realizarse en los dos pasos del método, tanto en el entrenamiento, incluyendo este la creación del descriptor y la búsqueda del hiperplano, como en el proceso de detección.

En el primer paso, los parámetros son configurados para obtener el clasificador con el mayor rendimiento y precisión, por ejemplo, es aquí donde se designa cuál será la ventana de detección o con qué margen vamos a permitir la detección de falsos positivos. Por otro lado, el según paso, o el proceso de detección, está más encaminado hacia la mejora de los resultados del clasificador ya obtenido, además de su configuración. Si el detector extraído del paso de entrenamiento no realiza correctamente su tarea y no es capaz de detectar correctamente los vehículos, aun intentando perfeccionar la búsqueda, esta no va a conseguir resultados positivos.

4.2.1 Entrenamiento

Como se ha ido introduciendo a lo largo de este capítulo, para poner en funcionamiento los entrenamientos era necesaria la correcta distribución de las imágenes en el proyecto, así como la creación de la carpeta correspondiente donde los archivos de datos creados se irían guardando.

Así en el proyecto deberían existir cuatro carpetas esenciales las cuales son:

- Pos: esta carpeta incluirá todas aquellas muestras positivas. En el código queda recogida la ruta de esta para el proceso de generación del descriptor HOG.

- Neg: de igual forma que las imágenes positivas, las negativas también tienen su directorio correspondiente. A su vez también es recogido en el código del programa.

- Genfiles: esta carpeta recogerá los archivos creados durante el entrenamiento, así como el archivo final del clasificador. Este último se guarda bajo el nombre *descriptorvector.dat* y es el que se introducirá en el proyecto de la detección.

- SVMLight: por ultimo nos encontramos con la carpeta que será destino de todos los archivos del SVMLight que hemos mencionado anteriormente.

Una vez concluidas esas tareas, el proyecto está preparado para empezar a entrenar a falta de la configuración de ciertos parámetros en el código. Estos pueden ser diferenciados según pertenezcan a la extracción de características para el HOG, a la búsqueda del plano de separación de los dos conjuntos o a otros parámetros, como el tamaño de las imágenes.

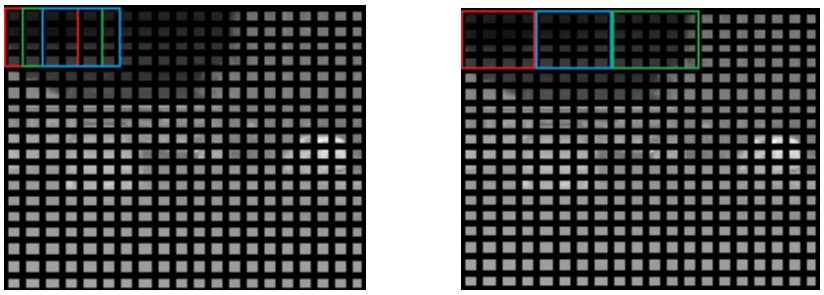
4.2.1.1 Características HOG

Estas serían algunas de las características relacionadas con el HOG, tanto para el entrenamiento como para la detección, como se verá posteriormente, ya que la función de detección también pertenece a esta clase. De esta forma, se han modificado los siguientes parámetros:

- Winsize: se refiere al tamaño de la ventana de HOG. Este está directamente relacionado con el tamaño de las imágenes, ya que no pueden ser distintos. Debe tomar valores múltiplos de 16, sin la necesidad de ser una ventana cuadrada. Por ejemplo, para la detección de peatones de Dalal y Triggs [3] se hizo uso de una ventana de 64x128 píxeles. Este también será el tamaño de ventana de detección del clasificador, es decir, una vez recorridas las imágenes de testeo, sólo se detectarán vehículos del tamaño del Winsize o mayores, pero nunca menores. Este es uno de los grandes inconvenientes de este tipo de métodos basados en una base de datos de imágenes, pero como se comentará posteriormente, se podrá solucionar en la parte de detección gracias al reescalado piramidal.

- Winstride: hace referencia a cómo nos desplazamos por la imagen en los distintos ejes. Supongamos una ventana de 64x64 píxeles y un Winstride de 1x1 píxeles, esto significa que nos movemos por la imagen con una ventana de 64x64 píxeles con un desplazamiento de la unidad tanto en el eje x como en el

eje y. Si por el contrario se eligiese un Winstride de 8x8 píxeles, los movimientos en cada eje serían de 8 pixeles, en vez de uno. Esto puede observarse en las figuras 4.5a y 4.5b, donde se exponen estos casos respectivamente.



(a) (b)

Figura 4.5: (a) Winstride 1x1 (b) Winstride 4x4

Este parámetro puede llegar a ofrecernos mucha información, si sus valores son reducidos, pero sin embargo también ralentizaría bastante el proceso. Es por ello que hay que encontrar el punto de equilibrio entre la precisión que estamos buscando frente al tiempo que puede estar realizando la tarea.

- Padding: gracias a este, se puede entrenar al clasificador para que sea capaz de detectar objetos que no se muestran completamente. Este añade píxeles en ambas direcciones con la finalidad de que, al situar una ventana sobre algunas características importantes que se encuentran en los bordes de la imagen, se pueda extraer esa información o en su defecto, detectar ese objeto. Cuanto menor sea el tamaño de este, mayor sea el tiempo de ejecución. Afecta directamente a la generación del descriptor HOG, y al añadir en cada ventana esos píxeles de más, se obtienen más características que podrán ser utilizadas posteriormente. Al igual que con el parámetro anterior, es necesario encontrar el equilibrio para un resultado bueno y eficiente.

4.2.1.2 SVMLight

- Tipo de entrenamiento

  Como se ha comentado anteriormente, el SVM está preparado para tratar con entrenamientos de clasificaciones y regresiones. Es por ello, que para la realización de los distintos entrenamientos era necesario establecer que nuestro sistema se iba a comportar como un clasificador. Sin embargo, se realizaron algunas pruebas con el método de regresión, y en relación a las funciones que debía desempeñar el sistema como detector, no existía ninguna diferencia de resultados. El método de regresión engloba lo que serían clasificaciones binarias.

- Verbosity level

  Este parámetro no afecta directamente al entrenamiento en sí. Esto es lo que nos permite regular la cantidad de información que el programa imprime por pantalla mientras está entrenando, ofreciendo así, según el nivel de información que se desee, hasta el detalle de la última operación realizada. Activar esta opción, o lo que es lo mismo subir el nivel de información no es recomendable, puesto que esta tarea ralentiza el proceso de entrenamiento, debido a la constante impresión por pantalla.

- C

  Es lo que se puede denominar como el parámetro de compensación entre los errores que pueda cometer el sistema entrenado, y el margen que se designe para el entrenamiento. Es el llamado error de castigo, puesto que controla cómo de estricta es la clasificación. Con él se puede controlar la influencia que cada vector de soporte tiene sobre el sistema final, lo que además supone un error compensatorio para mantener la estabilidad del clasificador. Es, como se ha comentado anteriormente, lo que permite pasar de un margen suave a uno más severo.

  Es un elemento crítico ya que un valor muy elevado puede suponer trabajar sobre un número muy alto de vectores de soporte, y por contraposición sobreentrenar el sistema. Sin embrago, un valor muy reducido puede ocasionar lo contrario y que el sistema no aprenda de las muestras. Un valor reducido dará lugar a numerosos errores en las detecciones, mientras que un valor elevado puede no llegar a detectar nada. En otras palabras, una *c* pequeña supone un margen suave, mientras que una *c* muy elevada da lugar a un margen muy estricto.

- Epsilon

  Es un parámetro que, por lo general, suele tener más influencia en los entrenamientos de regresión. Este, junto con el margen, evita, entre otras cosas, el sobreentrenamiento que ocurre en este tipo de sistemas. Su función es controlar la suavidad o la dureza de la clasificación. En concreto, este se encarga de la denominada zona de insensibilidad, la cual puede calificarse como margen de tolerancia, es decir, este parámetro controla los requisitos para clasificar el objeto, lo que en otras palabras supone la precisión del clasificador final. Un valor elevado supone más libertad en la clasificación y menos exactitud. Una representación en un espacio de dos dimensiones puede observarse en la figura 4.6.

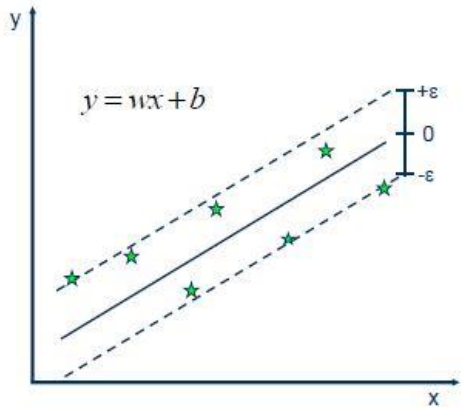


Figura 4.6: Epsilon [19]

- Rho

  Es el parámetro regulador que nos permite obtener planos ajustados (que pasen por el centro de coordenadas) o no ajustados (que no pasen por el centro de coordenadas). Esta es una práctica muy común para hacer más simple el procedimiento al partir de un punto conocido y muy recurrente. La relación matemática para de esto quedaría recogido por:

$$\sum_{i=1}^{n} \alpha_i h_i = 0$$

  La correspondencia en el código de programación seria definir el parámetro rho con un valor numérico contenido en el intervalo [0,1] según queramos el plano no ajustado o ajustado.

- Kernel type

  Este es el parámetro que permite los cambios de Kernel, como se explicó en la parte de fundamentos teóricos. Existen cinco tipos de Kernels posibles implementados, los cuales corresponden a un número de 0 a 4 respectivamente. Estos son lineales, polinómicos, radiales, tangenciales hiperbólicos, y la última opción te permite enlazarlos con un Kernel creado por el usuario mediante un archivo header. Por defecto tenemos el Kernel lineal, y será el que se utilizará durante todas las pruebas. Se realizaron pruebas con el polinomial y el radial, pero estos no permitían llegar al final del entrenamiento. Además, las respuestas que proporcionarían, no serían compatibles con el método de detección de HOG, *detectmultiscale*. Por este motivo, y ya que los resultados con el Kernel lineal era favorables, no se realizaron más pruebas al respecto.

Los parámetros restantes, que no han sido mencionados, están relacionados con los otros tipos de Kernel como es el caso de *d*, la cual controlaba el grado del polinomio,

en el Kernel polinomial, o *gamma*, que en el Kernel radial era considerado el parámetro libre en la función radial. A pesar de esto, en un sistema lineal como con el que trabajamos, con una buena optimización de los parámetros *c* y *epsilon*, se pueden obtener clasificadores bastante aceptables.

4.2.1.3 Otros parámetros a tener en cuenta

Esta sección no recoge parámetros como tal, sino más bien detalles o pequeños aspectos que se han modificado en este método. Podemos destacar 3, los cuales corresponden a las imágenes que se utilizan como base de datos.

El primer aspecto sería el tamaño de estas. Ya se ha comentado que el tamaño de la ventana del clasificador dependerá del tamaño del HOG que se seleccione, lo que equivale al tamaño de las imágenes. Sin embargo, además de ese detalle, el tamaño de la imagen es muy importante de cara a la calidad del resultado final. Puesto que estas, al extraer las características de HOG son estudiadas píxel a píxel, es bastante relevante que muestren exactamente el objeto a detectar. Cuanto mayor sea la muestra, mayor número de píxeles y por lo tanto mejor detalle del objeto, mientras a menor tamaño de la imagen, menor número de características. Esto fue modificado desde 32x32 píxeles hasta 64x64 píxeles para encontrar el tamaño idóneo.

El otro detalle tenido en cuenta fue la normalización de las imágenes. Estas se presentan en escalas de grises, y en algunas muestras existe muy poco contraste de iluminación. Es por ello que algunas de las pruebas se realizaron con la base de datos normalizada, aunque los resultados no fueron mencionablemente mejores.

El tercer aspecto a tener en cuenta fue la relación de imágenes positivas y negativas. Una relación coherente de información es lo que puede hacer que el detector funcione correctamente, detecte objetos, por falta de muestras negativas, o aprenda a no detectar nada, por culpa de un exceso de imágenes negativas. De esta forma, este fue otro de los parámetros a poner en el punto de mira para conseguir la final, y ansiada, configuración adecuada. Las relaciones probadas fueron 1:2.5, 1:4 y 1:40, siendo esto para cada imagen positiva 2, 4 ó 40 imágenes negativas.

4.2.2 Detección

Tras los entrenamientos, se obtiene el clasificador, que es el archivo *descrioptorvector.dat* que se ha mencionado anteriormente. Este es el encargado de clasificar correctamente un nuevo set de muestras, simulando el funcionamiento en el coche autónomo. Estas muestras son parte de esos videos recopilados, de los cuales se sacaron las del conjunto de entrenamiento. Estas nuevas secuencias no fueron utilizadas

para extraer ninguna información.

Así, gracias a esta segunda parte seremos capaces de estudiar y analizar el resultado y el impacto de esas modificaciones en el paso previo. Para ello se ha realizado un pequeño código de programación, el cual está en lenguaje C++, en el que se recoge la lectura del clasificador y la configuración de este, la lectura de las nuevas imágenes de testeo, así como la configuración de la función de detección, perteneciente a la clase HOG de la librería OpenCV, o la posterior detección y recopilación de información de este proceso.

4.2.2.1 Código de detección

El código de detección se puede dividir en cuatro partes significativas. A continuación se pasará por cada una de ellas explicando el algoritmo utilizado y el porqué de esa elección.

Configuración del clasificador HOG

La primera parte engloba la configuración del clasificador para que pueda ser utilizado como tal en la función de detección. Así, es necesario crear una variable de tipo HOG descriptor, la cual ya está definida en la clase HOG de la OpenCV. En esta variable se guarda la información del vector obtenido por el SVM. Para ello, es necesario primero cargar los datos en un vector, el cual posteriormente sea vinculado con la variable de HOG. Para que esto sea posible, se necesita saber la longitud del vector, el cual es proporcionado al final de cada entrenamiento, que cómo observamos en la parte teórica, era una variable muy sencilla de calcular.

Función de detección

En este apartado hablaremos de la función de detección, la cual es crucial para este sistema. Antes de nada es necesario cargar las imágenes de detección, por ello, gracias a una ruta predefinida, es posible hacer gracias a la función *imread*. Tras esto, y una vez configurado el HOG pasamos a la detección. Esta es llevaba a cabo por la función *detectmultiscale*, el cual es un método de la clase HOG, y por lo tanto es llamado desde nuestra variable. Esta cuenta con la entrada de la imagen donde detectar, el rectángulo donde guardar la información, que es un conjunto de puntos que representan la detección del vehículo, y parámetros de Threshold, escala o movimiento de la ventana deslizante.

Esta función y su configuración es lo que nos va a permitir incrementar o disminuir el número de falsos positivos, así como aumentar la velocidad de detección o disminuirla. Así esta depende de los siguientes parámetros:

- Imagen: como se ha comentado es el parámetro que le incluye la imagen de la cual se van a sacar las detecciones. Es el único parámetro obligatorio, junto con Rect. La imagen puede ser a color o en escala de grises, pero es conveniente que siga el mismo modelo que se utilizó para entrenar el clasificador.

- Rect: es un rectángulo que guarda la información de posición del objeto detectado, lo que en nuestro caso es con vehículos. Esto es posteriormente incluido en un vector de rectángulos, para poder trabajar con la información obtenida y extraerla.

- hitThreshold: es uno de los primeros parámetros a modificar de esta función con el fin de mejorar las detecciones. Este parámetro controla la distancia al plano de separación. Si esta distancia de las características es menor a la prefijada, esa detección se rechaza. En otras palabras, si la suma ponderada de los coeficientes de las características del vector de HOG es mayor que ese valor prefijado, entonces estamos ante una detección correcta. Esto lo que ofrece es más seguridad en la detección, y así, frente a un clasificador de margen suave, se descartan aquellas posibles muestras mal clasificadas.

- Winstride: como en el entrenamiento, es el parámetro que indica los pasos a tomar en cada eje con la ventana de desplazamiento. Este no tiene por qué ser igual que durante el entrenamiento. En esta parte juega un papel muy importante puesto que la correcta combinación de este y la escala, pueden dar lugar a una mejora en las detecciones. Sin embargo, al igual que el entrenamiento, cuando más pequeño sea el paso de avance, mayor tiempo se necesitara para recorrer la imagen. Esto es porque en cada avance, se calcula las características de HOG de esa ventana, y esto es un proceso que tiene un gran coste computacional. Es por ello que cuantas menos características se calculen, lo que significa un mayor paso de avance, menos tiempo se necesitará.

- Padding: tiene la misma función que durante el entrenamiento. Añadir algunos píxeles a la imagen por ambos ejes en la detección, según el artículo del Dalal y Triggs [3], mejora los resultados del clasificador. Sin embargo, también afecta negativamente al tiempo de procesado.

- Escala: es sin duda el parámetro más importante de la parte de detección, no sólo por los resultados que es capaz de conseguir, sino por el tiempo de cómputo que la modificación de su valor implica. Como se ha comentado, el clasificador es entrenado a un tamaño especificado, lo que da lugar a que solo realice detecciones que se encuentran a ese tamaño. Sin embargo en las imágenes esto no suele ocurrir, puesto que por ejemplo, en nuestro caso, existen coches a un tamaño mayor y menor que los tamaños prefijados. Es por ello que se realiza un

reescalado de la imagen para poder solventar este problema.

Con este método únicamente se realiza la transformación de la imagen para detectar objetos que se encuentran a un tamaño superior al del clasificador. El método de HOG, por sí mismo no es capaz de detectar objetos a tamaños inferiores. El procedimiento es simple, manteniendo constante la ventana de detección, se disminuye la imagen origen un factor que se desee. De esta forma podremos encontrar vehículos más grandes puesto que la imagen es más pequeña. Esto se realiza tantas veces como estipulado quede en el parámetro que se modifica. En la figura 4.7 se muestra como es el procedimiento:

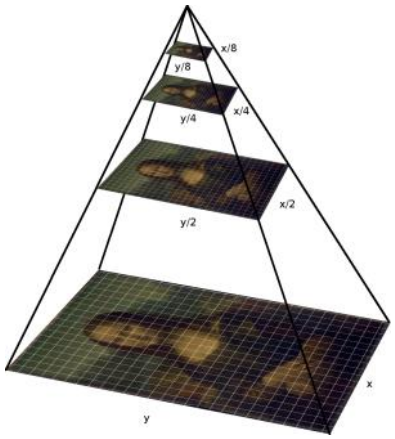


Figura 4.7: Proceso de reescalado [20]

El parámetro de la escala controla ese reescalado de la imagen, designando así mayor o menor número de capas, de forma que un menor valor de la escala aumentará el número de capas y por tanto, las posibles detecciones, y un mayor número de escala disminuirá el número de capas. Sin embargo, hay que volver a tener en cuenta el tiempo de cómputo, cuando menor sea la escala, mayor tiempo se necesitaría para analizar la imagen.

El valor de este parámetro ronda entre 1.01 y 1.5, y como se ha mencionado, en combinación con el paso de avance, se pueden conseguir resultados muy aceptables.

- finalThreshold: este parámetro controla las detecciones que pueden centrarse en torno a un objeto. Esto es que la mayoría de las ocasiones no existe una única detección, si no que varias son consideradas en posiciones muy cercanas. Este parámetro controla esas detecciones, y termina mostrando una única detección en ese objeto dependiendo de la distancia que exista entre las detecciones. Este fenómeno puede observarse en la figura 4.8. Este parámetro también es llamado groupThreshold, ya que se centra en los grupos de detecciones. Toma valores enteros si no trabaja junto con el parámetro de a continuación.

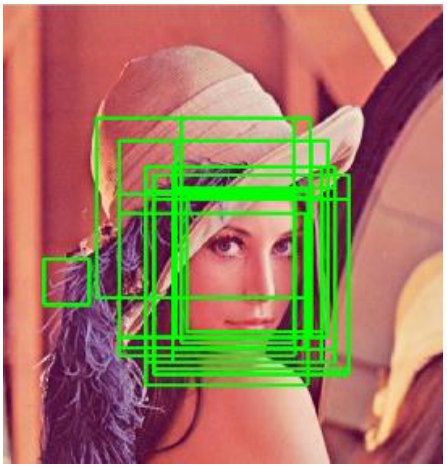


Figura 4.8: Numerosas detecciones [21]

- Mean-shift: por último, nos encontramos con una variable booleana, la cual nos permite activar o desactivar el método de mean-shift para los grupos de detecciones en los mismos objetos. La tarea final puede confundirse con el anterior, pero la combinación de ambos permite la gran precisión en las detecciones en grupo. Sus valores pueden ser true, si queremos aplicarlo, o false si por el contrario preferimos no trabajar con él.

Filtrados

En este apartado trataremos de explicar el proceso llevado a cabo, junto a la función de detección, con el fin de conseguir unos mejores resultados. Estos métodos, o filtrados tal y como lo hemos llamado, tratan de eliminar, en la medida de lo posible, todos aquellos falsos positivos localizables. Han sido utilizados tanto en el método de HOG, como en el de Haar, por lo que posteriormente no volverán a ser explicados.

Denominamos falsos positivos localizables a aquellos que son fácilmente distinguibles de los aleatorios, aquellos que siguen un patrón. Estos, por ejemplo, son aquellos ocasionados por la confusión de una nube o una farola con un vehículo, lo cual es muy común, o los falsos positivos que se encuentran en zonas donde no existirían coches, así como aquellos que pueden considerarse detecciones buenas, pero redundantes. Este es el caso de las, en términos generales, numerosas detecciones que suelen existir en torno a los vehículos, lo cual lejos de ser un problema, nos indica que el detector está funcionando correctamente y localiza el objeto en cuestión. Sin embargo, es necesario convertir todas esas detecciones en una única.

Por último, además de por la eliminación de las detecciones erróneas, este apartado recogerá cual es el método utilizado para poder reconocer aquellos vehículos que se

encuentran a una distancia lejana y por lo tanto son demasiado pequeños para ser reconocidos directamente por el HOG entrenado, puesto que recordamos, este no es capaz de detectar objetos de dimensiones inferiores al tamaño de entrenamiento, pero si superiores.

Se pasará a explicar el filtro de tamaño, que se encarga de la última aplicación mencionada, el de las localizaciones, según la zona de la imagen y el Non Maxima Supression, que se encargará de las numerosas detecciones en un mismo vehículo. Cabe mencionar que este apartado es común para ambos métodos de entrenamiento, HOG y Haar, y posteriormente se hará alusión a ellos en los apartados de resultados.

Filtro de tamaño

Como hemos comprobado, con esta función somos capaces de detectar objetos del mismo tamaño o mayores que nuestro clasificador. Sin embargo, en nuestro caso también existen vehículos que se encuentran a mayor distancia de la cámara y por lo tanto a menor tamaño. Para solventar este problema se han añadido unas líneas al código que hacen el mismo funcionamiento de la escala, en la función anterior, pero de forma inversa. Si para detectar coches más grandes hacía falta disminuir la imagen, para detectar coches más pequeños es necesario aumentar la imagen.

En este caso, se vuelve a pasar la función *detectmultiscale* y se guarda la información. Como es posible que alguna de estas detecciones coincida con las realizadas a tamaño original, se ha incluido un filtro que permite desechar aquellas que ya han sido localizadas.

Se ha utilizado una escala de 1/3, para conseguir, con el mayor HOG, el de 64x64 píxeles, detectar coches que están a más de 15 metros de longitud. Así, una vez las detecciones son realizadas, la información recopilada es devuelta a la escala original para poder trabajar con ella en el vector general de rectángulos, o detecciones,

Filtro de localización

Por las limitaciones geométricas con las que trabajaremos en nuestra aplicación, como por ejemplo que los vehículos en las secuencias tomadas por la cámara estarán situados en una altura media de la imagen, lo que significa que las zonas altas, lo que normalmente suele corresponder a cielo, y las zonas bajas, que por lo general pertenece a carretera, son regiones donde el detector no debería buscar coches, o al menos no detectarlos. Es por ello que se creó este filtro, el cual queda representado por la figura 4.9.

Frente a los posible falsos positivos que nos podemos encontrar, se decidió limitar el

espacio de detección a una zona donde existiese la posibilidad de haber un coche. Para ello, tras recorrer la imagen con el *detectmultiscale*, se analizaría la situación de la esquina superior izquierda del rectángulo detectado. Si esta se encontraba en la franja permitida sería considerado detección, mientras que de lo contrario, seria desechado.

Figura 4.9: Límites geográficos

Además de esto, se ha analizado la imagen y se ha establecido unos umbrales de posibles localizaciones de los vehículos. Esto nos ha permitido eliminar gran número de falsos positivos, que se encontraban dentro de la zona de detección, pero que eran fácilmente localizables. El algoritmo consiste en establecer unos límites según el área y la posición del rectángulo en la imagen. Así, vehículos más grandes podrán tener su esquina superior izquierda en zonas elevadas, como vehículos pequeños no podrán pasar de un mínimo establecido. En el caso de los vehículos más cercanos, y por lo tanto a mayor tamaño, pueden infringir la zona de detección, si su área es mayor a un umbral preestablecido, como se muestra en la figura 4.10.

Figura 4.10: Excepción límite geográfico

Non Maxima Supression

Como ya se ha comentado es el método que elimina las detenciones sobrantes en un verdadero positivo. Estas por lo general suelen tener una apariencia similar a la de las imágenes de a continuación. Gran número de cuadrados en distintas posiciones, cercanas entre ellas, alrededor del objeto.

Este método, basado en un algoritmo inicial que se encuentra en [22], y que fue modificado por Nuzhny007, miembro del portal de Github, para implementarlo en C++. La fuente se encuentra en [23]. Además, para nuestra aplicación, este se ha modificado con el fin de otorgarle más características y poder concretar más la eliminación de esos falsos positivos.

El código se basa en recopilar todos los rectángulos, detecciones, que hay en la imagen, y ordenarlos por la altura de la coordenada de la esquina inferior derecha de cada uno de ellos, como puede verse en la figura 4.11, y lo que es denominado, en la clase Rectángulo de C++, br ().y. A partir de ahí, partiendo de aquel cuyo valor sea mayor, y verificar si este tiene un área común con otros rectángulos superior a un valor establecido. Esto se hace en base a una relación de áreas, entre la unión de la que ocupan por separado y la que ocupan conjuntamente, o lo que es lo mismo, el área que comparten. Si esta relación es mayor al umbral predefinido, son encontramos ante un caso en el que hay que aplicar este método.

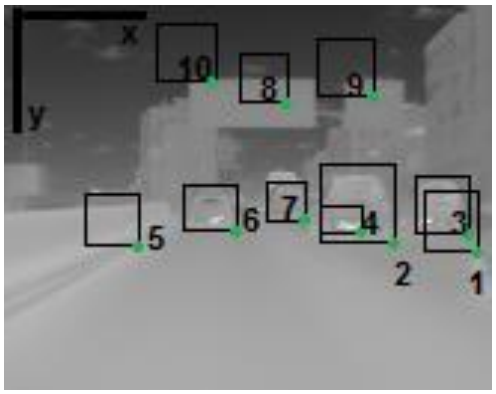


Figura 4.11: Orden por esquina inferior

La aplicación de este método tendrá distintos resultados según las áreas y las posiciones de los rectángulos, una vez cumplida la condición del umbral. Esto no estaba implementado en el código original, pero tras la observación de los resultados, nos dimos cuenta que se eliminaban las detecciones correctas, puesto que la única condición de elección era la posición del rectángulo, teniendo preferencia aquellos que se encontraban en las zonas inferiores de las imágenes.

Es por ello que se desarrolló un pequeño código en el cual, las condiciones estaban basadas en el peso que tenían unas áreas sobre otras, cuál era el tamaño de estas o en que zonas de la imagen se encontraban. Se trataría de dar preferencia a aquellas detecciones situadas en una zona alta de la imagen, y que además tuviesen un área mayor. Contando con que denominamos rectángulo 1, a aquel que tiene el valor de la coordenada de la y mayor que todos demás, y rectángulo 2 aquel que le sigue en la lista. De esta forma contábamos con las siguientes normas:

- Si el rectángulo 1 es mayor que el 2, se estudiará además cuál es la diferencia de las áreas y cuál es la posición de la esquina superior derecha. Si la diferencia es inferior a 1000 píxeles^2 entonces se comprobará la esquina, y si esta se encuentra en una posición más elevada, nos quedaremos con el rectángulo 1, puesto que muestra que es más grande y, al empezar en una zona más elevada, puede englobar más imagen. Si por el contrario, la diferencia de áreas es menor que 1000 píxeles^2, y la esquina superior del rectángulo 1 se encuentra en una zona inferior a la del otro a analizar, nos quedaremos con el segundo rectángulo, quedando aquí clara la preferencia ante zonas superiores. Si la diferencia de áreas es superior a 1000 píxeles^2, directamente nos quedamos con el primer rectángulo, ya que engloba más información. Esto puede observarse en las secuencias de la figura 4.12, respectivamente.

Figura 4.12: Respuestas de relevancia

- Si el área del rectángulo 2 es superior a la del 1, tenemos que tener en cuenta como de grande es el área y donde se encuentra situada. En algunas ocasiones existen falsos positivos en el cielo, que por su posición y sus tamaño no pueden cumplir las condiciones de vehículos. Esto también es recogido en esta parte del código limitando a aquellas detecciones con un área inferior a 15000 píxeles^2, que tengan su esquina superior por debajo de y=50. Esto es, porque solo aquellos automóviles que se encuentren a una distancia mediana, tendrán su detección por esas zonas, y estos, tras analizar la base de datos, cumplen las características de tener áreas mayores a ese valor. Esto queda ilustrado en la figura 4.13.

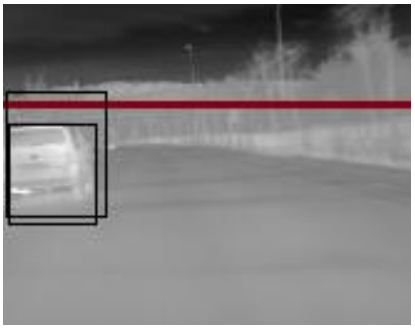


Figura 4.13: Excepción geográfica

4.2.3 Resultados

Para la fase de detección, en ambos métodos, se ha recurrido al análisis y estudio de los distintos parámetros gracias a pruebas visuales, como las imágenes que se irán mostrando en cada caso, así como, algunas gráficas de rendimiento o de tiempo de cómputo. Para el primer caso, se estudiaran los parámetros de *precision*, y *recall*:

$$\boldsymbol{Precision} = \frac{\boldsymbol{Verdaderos\ positivos}}{\boldsymbol{Verdaderos\ positivos + Falsos\ positivos}} \tag{4.1}$$

$$\boldsymbol{Recall} = \frac{\boldsymbol{Verdaderos\ positivos}}{\boldsymbol{Verdaderos\ positivos + Falsos\ negativos}} \tag{4.2}$$

Esto será calculado para los parámetros más relevantes de cada método, así como para la configuración final elegida para cada uno, gracias a la cual se podrá elegir el clasificador más óptimo.

4.2.3.1 Análisis de los parámetros de entrenamiento

A continuación se explicarán las distintas elecciones de parámetros a lo largo de los distintos entrenamientos, hasta conseguir el detector más óptimo. Se partirán de unos valores por defecto que, al ir conociendo su valor correcto, serán modificados. Es decir, en la comparación de parámetros, los entrenamientos serán los mismos a excepción de ese parámetro a modificar.

Número de imágenes

Uno de los primeros aspectos a tener en cuenta fue la relación entre imágenes positivas y negativas. Esta relación depende de las imágenes de entrada que se tengan, así como de las características que estas ofrezcan, por lo que dependiendo del objeto a estudiar y la calidad de las imágenes de partida, será una relación u otra.

El número de imágenes positivas era constante, puesto que no existían más pruebas de las que poder sacar recortes. Así, el número de imágenes que se cambiaría, con el fin de encontrar la relación perfecta, sería el de negativas. De esta forma, se realizaron distintas pruebas con un tamaño de 64x64 píxeles, puesto que a mayor tamaño, mayor número de características y por lo tanto, mejores resultados.

Así, comenzamos con una relación en torno a 1:1, obteníamos que las detecciones eran pobres. Existían numerosos falsos positivos así como falsos negativos. Esto puede verse en la figura 4.14.

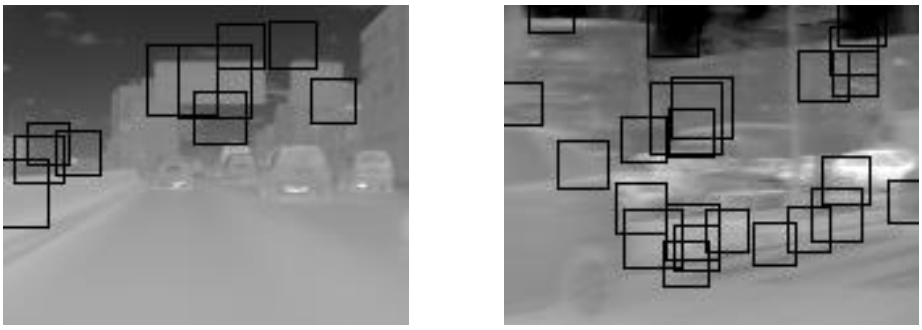


Figura 4.14: Positivas-Negativas 1:1

Frente a esto se aumentó considerablemente el número de muestras negativas, colocando esta vez en torno a 40000 imágenes. Esto, a pesar de mejorar levemente los resultados en comparación con la primera prueba, no proporcionó un detector con el que se pudiesen conseguir los objetivos deseados. Los falsos positivos podían considerarse aleatorios, y las detecciones de vehículos no se ajustaban a los que se había proporcionado en los entrenamientos.

El tamaño de la ventana, y por lo tanto el tamaño de las imágenes, tuvo que ser reducido, puesto que debido al número tan elevado de muestras, los entrenamientos eran extremadamente largos, llegando a sobrepasar las 72 horas, no llegando, en algunas ocasiones a terminar su tarea. Bajando el tamaño a 48x48 píxeles, se obtuvieron los resultados que se muestran en la figura 4.15, donde se puede observar lo que se ha comentado anteriormente.

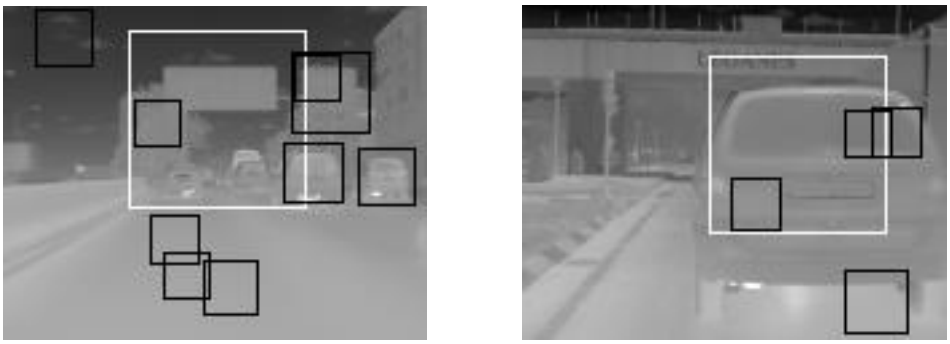


Figura 4.15: Positivas-Negativas 1:40

Tras estos se realizaron algunas pruebas más para establecer un valor límite de muestras, y tras esto llegamos a la solución más óptima. Para esta, se aumentó el número de negativas a unas 4000, lo que indicaba una relación de 1:2,5. En este caso los resultados mejoraron considerablemente, reduciéndose tanto los falsos positivos como los negativos. Los vehículos eran mejor ajustados en las detecciones y aquellos falsos positivos existentes eran más localizables. Estas localizaciones solían ser en el cielo confundiendo las nubes, los paneles de información o las farolas con las estructuras de los vehículos.

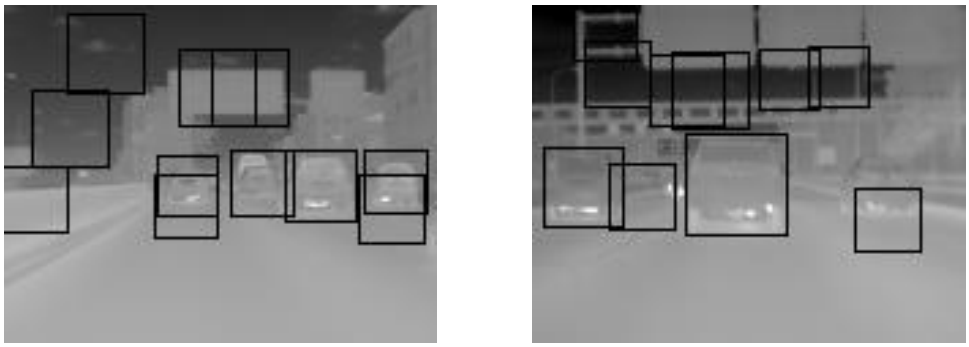


Figura 4.16: Positivas-Negativas 1:2.5

Finalmente se decidió seguir con esta última relación de imágenes, con el propósito de conseguir configurar todos los parámetros restantes, y así, poder eliminar esos falsos positivos presentes en las detecciones.

Tamaño de imágenes

Partiendo de una relación de positivas negativas 1:2.5, para el tamaño de las imágenes es necesario estudiar la base de datos de la que partimos. Teníamos 3 posibilidades iniciales según el tamaño de las secuencias tomadas. Tras los recortes del etiquetado, pudimos observar cómo se podían organizar los vehículos en tres grupos según el tamaño de los vehículos. Esto serviría para designar el tamaño del HOG, utilizando así aquel más común entre las muestras.

Entre 10x10 píxeles y 30x30 píxeles, era el grupo más abundante. Estos vehículos eran los que se encontraban a una distancia alejada de la cámara. Estos se transformaron a un tamaño de 32x32 píxeles. Esto significaba que el tamaño de la ventana del HOG era muy pequeño, y por lo tanto, el número de características utilizadas para los entrenamientos iba a ser escaso, y por lo tanto poco preciso. Es por ello que la idea inicial de guiarnos por el grupo más abundante fue descartada.

Esto puede verse reflejado en la figura 4.17. Vemos como a pesar de intentar buscar los vehículos, sigue cometiendo muchos falsos positivos y falsos negativos. Además, cabe mencionar que, aunque los entrenamientos sean más rápidos, puesto que trabaja con menos información, en la detección el tiempo requerido es excesivo, no pudiéndose cumplir así la condición de trabajo en tiempo real. Esto es debido a que la ventana de HOG es menor, y por lo tanto necesita más tiempo para cubrir toda la imagen.

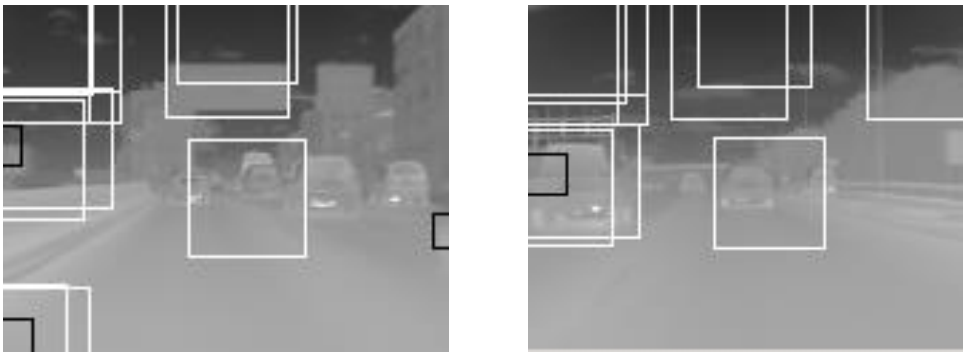


Figura 4.17: Tamaño 32x32 píxeles

Entre 30x30 píxeles y 50x50 píxeles, los cuales fueron traducidos a 48x48 píxeles. Era el grupo de los coches a una distancia media, y era el más escaso. Esto significaba que habría un mayor número de imágenes que serían reescaladas considerablemente, y esto, lo que empeoraría la calidad de las imágenes, y a su vez, los resultados, como se observa en la figura 4.18. Es por ello que en comparación con el anterior, los resultados eran sensiblemente peores. El tiempo de procesamiento si se redujo levemente, pero no lo suficiente para poder trabajar en tiempo real. No se aportaban las suficientes características válidas para que se realizasen detecciones correctas.

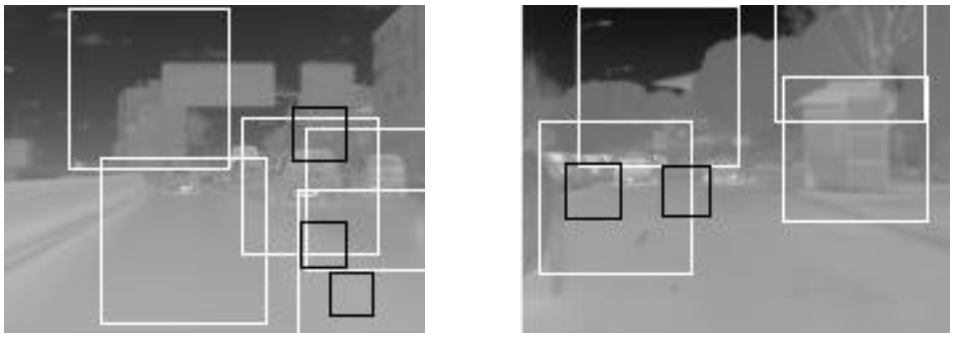


Figura 4.18: Tamaño 48x48 píxeles

Entre 50x50 píxeles hasta 100x100 píxeles: era el que representaba aquellos vehículos que se encontraban a una distancia cercana media. Al igual que el grupo anterior, no era muy abundante, pero al obtener mayores recortes, las características obtenidas aumentaban y por lo tanto la precisión del detector cambiaba considerablemente.

Los entrenamientos eran más largos, y tenían un mayor gasto computacional, pero el sistema de detección era veloz y permitía una aplicación a tiempo real. Además los resultados eran claramente mejores que los anteriores ejemplos, lo que llevo a elegir este tamaño como el definitivo. No existe un gran número de falsos negativos, a excepción de aquellos coches que por su diminuto tamaño no se pueden detectar, como se comentará posteriormente, y los falsos positivos, por lo general, pueden ser modificados por los métodos de Non Maxima Supression o por la limitación de zonas. El resultado se recoge en la figura 4.19 y se observa una clara mejoría en la detección.

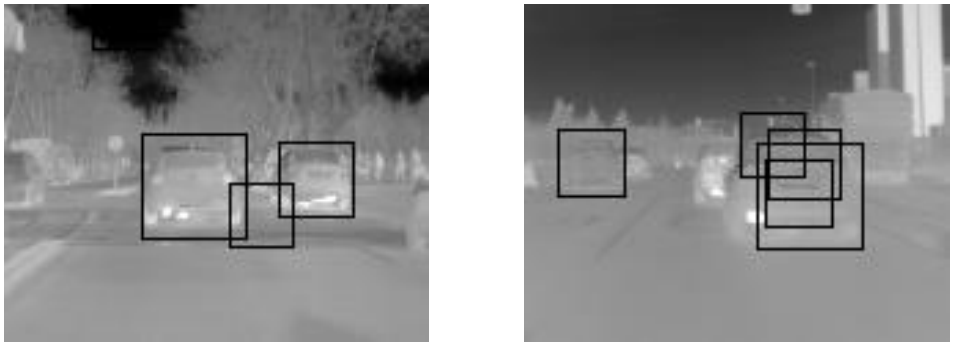


Figura 4.19: Tamaño 64x64 píxeles

Estos resultados pueden apreciarse claramente en la figura 4.20, la cual es una gráfica que recoge los valores de *precision* y *recall* para cada tamaño de la imagen. Observamos como los resultados ya estudiados corresponden con los datos numéricos obtenidos. Para el caso del tamaño mayor, el cual será con el que sigamos el entrenamiento, contamos con una *precision* de un 47% y un *recall* en torno al 65%.

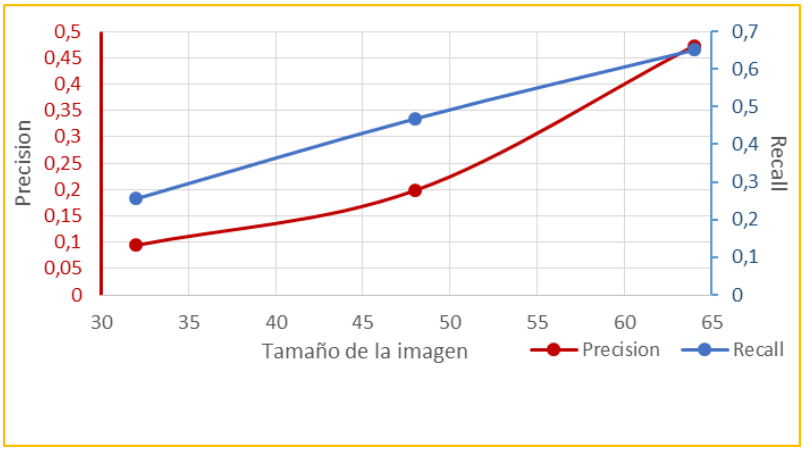


Figura 4.20: Precision-Recall Tamaño de la imagen

Winstride

Una vez designada la base de datos, y el tamaño de las imágenes que la componen, se pasará a estudiar cómo se analizan las imágenes para obtener las características de HOG. Para ello, los dos parámetros principales son el *winstride* y el *padding*. En este caso, el winstride, como se había comentado, es la ventana con la que nos desplazamos en la imagen para buscar las características.

Así, partiendo de una ventana 4x4 píxeles, se puede observar en la figura 4.21 como los resultados son bastante favorables, sin embargo el tiempo de cómputo era muy elevado, ya que al desplazarnos con una ventana pequeña, es necesario un mayor número de movimientos.

Figura 4.21: Winstride 4x4 píxeles

Subiendo el tamaño de ventana a una de 8x8 píxeles, se aprecia en la figura 4.22 como los resultados mejoran levemente, pero sin embargo el sistema ha tardado mucho menos en realizar el entrenamiento. Esto es a causa de lo comentado anteriormente. Al trabajar con una ventana más grande, el número de desplazamientos será menor.

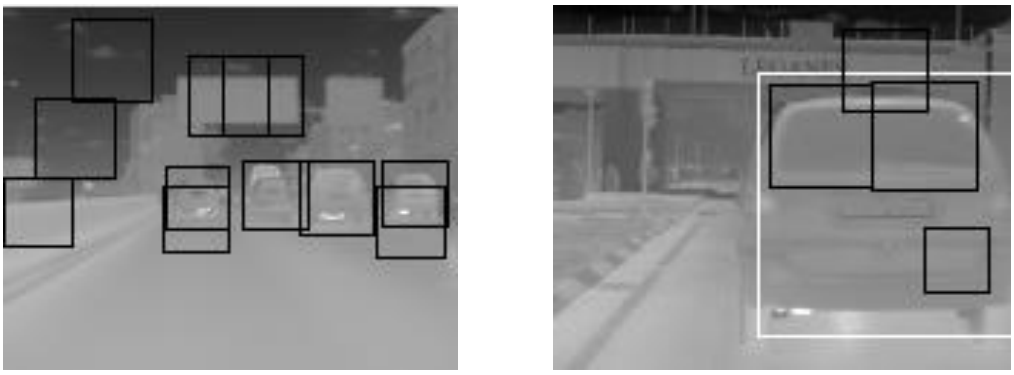


Figura 4.22: Winstride 8x8 píxeles

Por último, si se aumenta el tamaño de la ventana a una de 16x16 píxeles, los resultados empeoran levemente, apareciendo un mayor número de falsos positivos en zonas de detección, y el tiempo de cómputo tampoco se ven tan reducido con respecto a la prueba anterior. Esto se muestra en la imagen 4.23.

Figura 4.23: Winstride 16x16 píxeles

Tras esto, la elección del tamaño de la ventana de trabajo se basó, prácticamente, en el

tiempo de cómputo, ya que los resultados de las detecciones diferían bastante poco entre pruebas. Es por ello que el valor que este parámetro que se tomó para futuros entrenamientos fue el de una ventana de 8x8 píxeles, debido a su equilibrio entre extracción de características y velocidad de procesamiento.

Padding

Como se ha mencionado, este parámetro es interesante de cara al desarrollo del entrenamiento por los resultados que esto genera, pero sobre todo por el tiempo que necesita para ello. Como ya sabemos, aumentar este valor, lo que permite es añadir píxeles en la ventana de trabajo para poder estudiar bien las características, sobre todo para aquellos objetos que están en los bordes de la imagen. Sin embargo, esto tiene un gasto computacional muy grande, lo que no nos permitió terminar ningún entrenamiento con un padding de 4x4 píxeles. Es por ello que no se realizaron pruebas con esta configuración y directamente se definió este parámetros con el valor por defecto, 0x0 píxeles, nos ofertaba buenos resultados.

Margen

El margen es uno de los parámetros más importantes, puesto que en base a lo que se establezca su valor se tendrán unos resultados positivos o negativos. Esto es lo que nos permite jugar con los falsos positivos y negativos que nos encontremos en las imágenes.

El valor por defecto que nos encontramos es de c=0.01, lo que implica un margen muy suave. Esto es la permisión de numerosas excepciones que da lugar a falsos positivos. El tiempo de cómputo es muy corto, y los resultados se pueden apreciar en la figura 4.24.

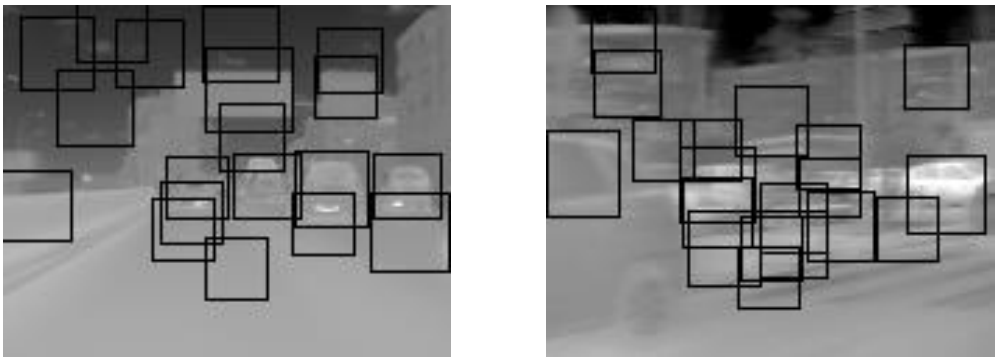


Figura 4.24: Margen 0.01

Como se puede observar, las imágenes cuentan con un gran número de falsos positivos, aunque también son capaces de detectar correctamente los vehículos existentes, elevando así el número de verdaderos positivos. Para mejorar el resultado, necesitamos endurecer el margen, controlando las excepciones que dejamos pasar. Si se aumenta el

valor a 0.1, como se recoge en la figura 4.25, se observa como los resultados han mejorado considerablemente. Un detalle a tener en cuenta es que el tiempo de cómputo es mayor que en la prueba anterior.

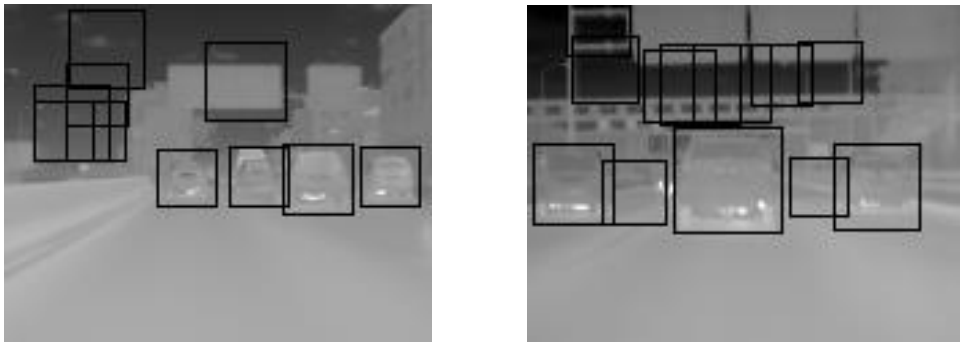


Figura 4.25: Margen 0.1

A pesar de haber mejorado los resultados, siguen existiendo todavía numerosos falsos positivos. Por ello se volvió a aumentar el valor de la *c* a la unidad y se consiguieron los resultados más óptimos de las pruebas realizadas. En la figura 4.26 se observa como el número de falsos positivos ha sido reducido, y las detecciones siguen siendo favorables.

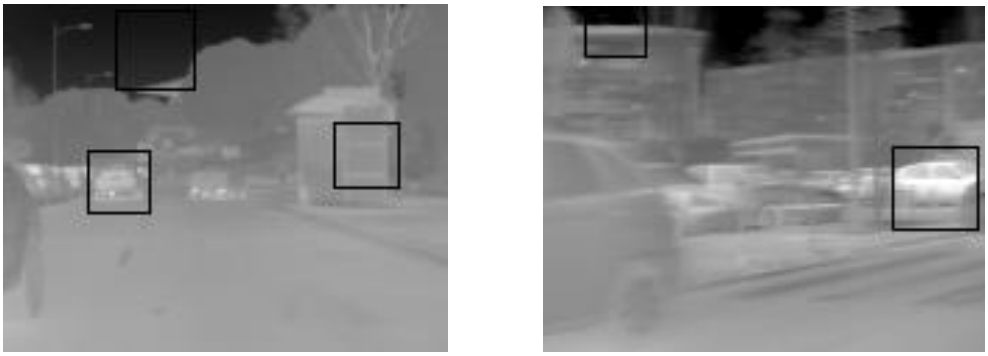


Figura 4.26: Margen 1

Con una *precision* algo inferior a lo que se consigue con las primeras dos pruebas, este último valor nos permite encontrar el equilibrio entre detecciones y errores. El número de falsos positivos disminuye, y además, estos se localizan fácilmente, lo que ayuda a su eliminación. El tiempo empleado para conseguir estos resultados aumenta notablemente, dando lugar a entrenamientos de varias horas.

La comparativa entre los tres valores queda recogida en la figura 4.27, donde se establecen las relaciones del *precision* y *recall* frente a la modificación del margen.

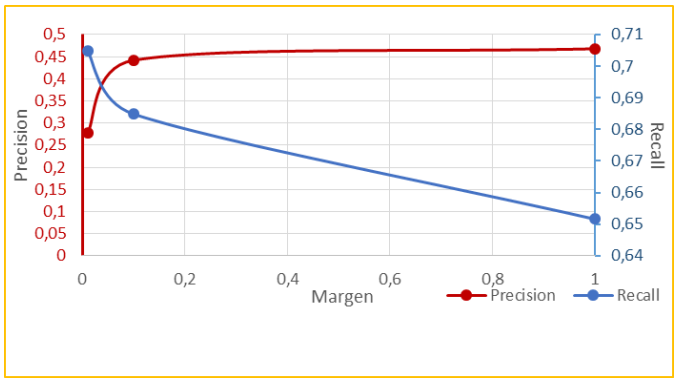


Figura 4.27: Precision-Recall Margen

Se probó a definir el margen con un valor de 10, con el fin de eliminar el número máximo de falsos positivos posible, pero no se consiguió. Los resultados fueron muy similares al anterior, y el tiempo de cómputo fue muy elevado, llegando a durar un entrenamiento más de 48 horas. Es por ello, que el valor óptimo para este parámetro es la unidad, c=1, de forma que con las siguientes variables podamos afinar la detección.

Epsilon

De igual forma que el anterior, y partiendo de un margen de la unidad, tras las pruebas realizadas podemos observar que cuanto mayor es la épsilon con la que estamos trabajando, más estricta es la detección, y por lo tanto menos detecciones obtenemos. Así, para el valor por defecto de 0.1 observamos, según la figura 4.28, como los resultados son bastante óptimos.

Figura 4.28: Epsilon 0.1

Si se disminuye este valor la detección se hace menos estricta, apareciendo un gran número de falsos positivos y obteniendo resultados como los de la figura 4.19, con una épsilon de 0.001:

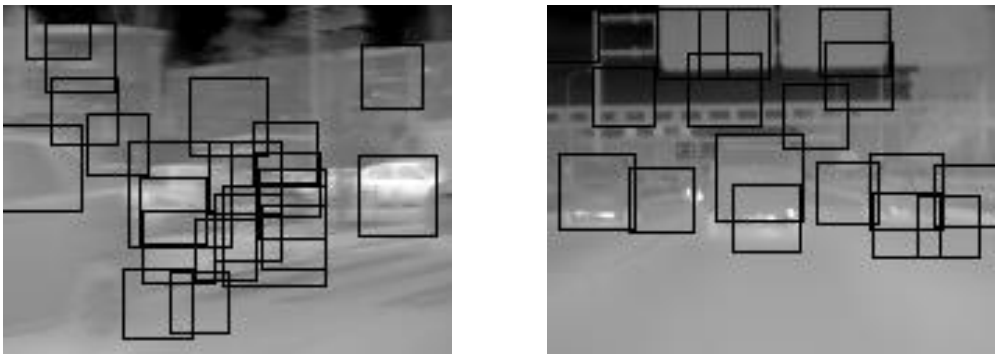


Figura 4.29: Epsilon 0.001

Si por el contrario, se aumenta esta variable a la unidad, la detección se vuelve tan estricta que no se consigue clasificar nada, devolviéndonos el programa las imágenes de entrada. Esto se observa en la figura 4.30.

Figura 4.30: Epsilon 1

Tras esto, podemos concluir que la mejor combinación del margen y épsilon es la de la unidad y 0.1 respectivamente, consiguiendo resultados muy favorables.

Rho

Por último, tenemos Rho como el último parámetro de entrenamiento a modificar. Este, que permitía el crear planos ajustados al origen de coordenadas, y puede tomar valores comprendidos entre 0 y 1, siendo este último el correspondiente al ajustado, y el que se encuentra por defecto. Los resultados se muestran en las figuras 4.31 y 4.32, para el plano ajustado y no ajustado respectivamente.

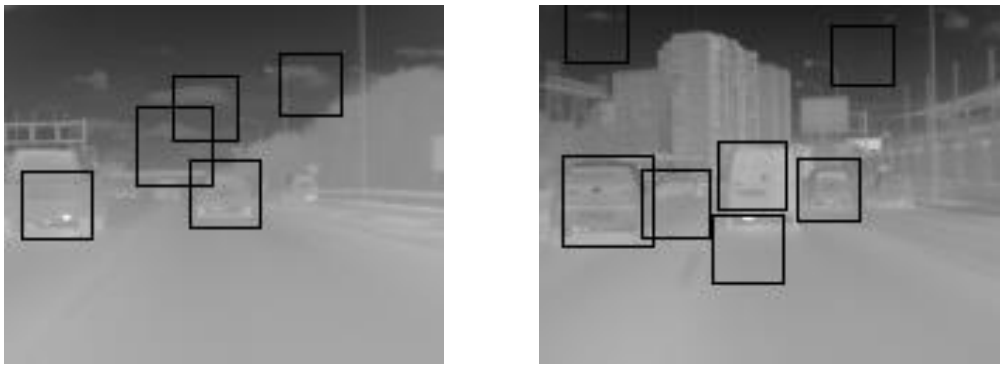


Figura 4.31: Plano ajustado -- 1

Se puede observar como las detecciones son seguras, existiendo algunos falsos positivos en las imágenes. Sin embargo, al modificar el parámetro y desajusta el plano, los resultados empeoran notablemente. Esto es debido a la complejidad matemática que supone la clasificación al no pasar por el origen de coordenadas.

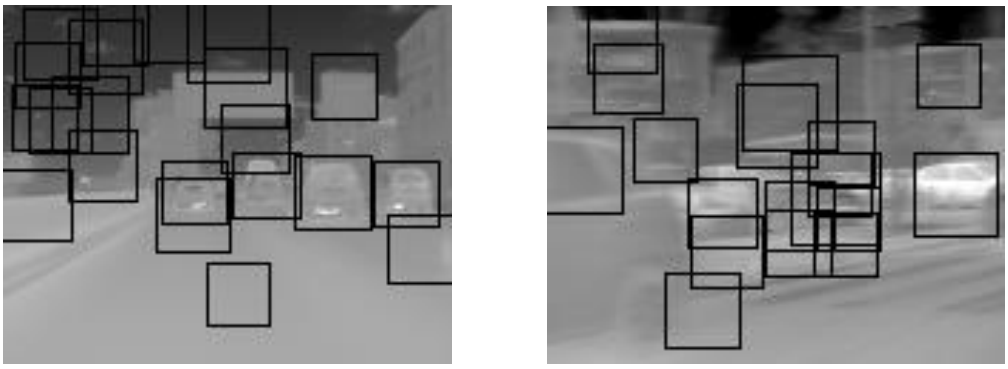


Figura 4.32: Plano desajustado - 0.01

De esta forma podemos concluir con la elección del plano ajustado, y por lo tanto del valor de la unidad para este parámetro. Es por ello que la configuración final óptima de salida de los entrenamientos es la recogida por los parámetros:

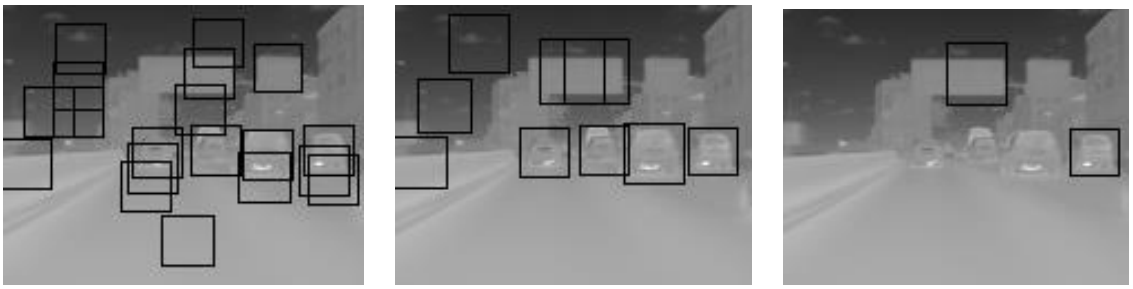
$$\boldsymbol{Relación\ de\ imágenes: 1:2.5}$$
$$\boldsymbol{Tamaño: 64x64\ píxeles}$$
$$\boldsymbol{Winstride: 8x8\ píxeles}$$
$$\boldsymbol{Padding: 0x0\ píxeles}$$
$$\boldsymbol{Margen: 1}$$
$$\boldsymbol{Epsilon: 0.1}$$
$$\boldsymbol{Rho: 1.0}$$

4.2.3.2 Análisis de los parámetros de detección

Por orden de aparición en la función, vamos a estudiar los parámetros que componen la función de detección, *detectmultiscale*.

HitThreshold

Como se ha comentado es el parámetro que controla la distancia de las características al plano de separación. Es un valor límite que nos permite filtrar aquellos posibles falsos positivos. Por esta razón, cuando mayor sea la distancia más permisiva será la detección, mientras que, al aumentar este valor, proporcionalmente irá siendo menor el número de detenciones realizadas por el clasificador. Esto puede eliminar tanto los falsos positivos, lo cual es nuestro objetivo, como verdaderos positivos, lo que nos aleja de buenos resultados. Así, podemos observar en las figuras 4.33ª, 4.33b y 4.33c, las cuales corresponden a un hitThreshold de 0.001, 0.05 y 0.3, respectivamente, como este parámetro influye en las detecciones de vehículos.



(a) (b) (c)

Figura 4.33:(a) hitThreshold 0.001 (b) hitThreshold 0.05 (c) hitThreshold 0.3

Tras esto, podemos observar como el mejor valor, a pesar de seguir contando con falsos positivos, es el que corresponde a 0.05. La última prueba elimina muchos de los falsos positivos de los que hablamos, pero sin embargo, también elimina correctas detecciones de vehículos.

Winstride

En cuanto al tamaño del a ventana, vuelve a jugar el mismo papel que antes, y se trata de encontrar el equilibrio entre lo bueno que ofrece y lo malo que esto supone. Se ha vuelto a estudiar los tamaños de 4x4 píxeles, 8x8 píxeles y 16x16 píxeles, y se han obtenido unas conclusiones muy similares a las del entrenamiento. Los resultados de cada tamaño se encuentran en las figuras 4.34, 4.35 y 4.36 respectivamente.

Con la ventana más pequeña, por lo general, se estudian a fondo más características, y por lo tanto es algo más preciso, pero no de gran importancia. En cuanto al tamaño mediano, parece que vuelve a ser el que ofrece un mejor comportamiento según su relación comportamiento tiempo.

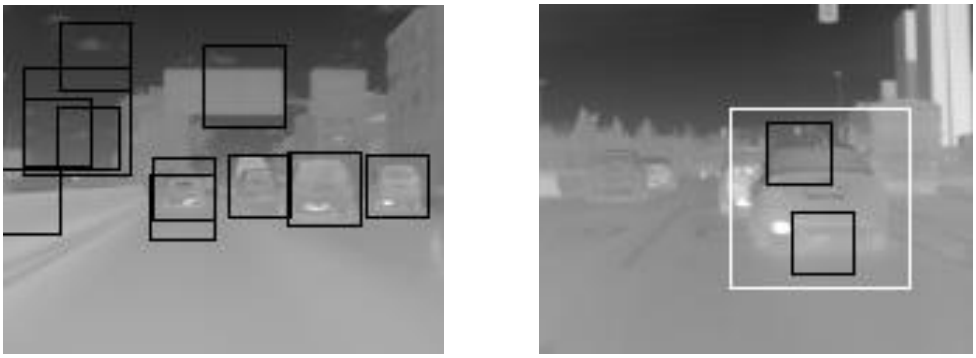


Figura 4.34: Winstride 4x4 píxeles

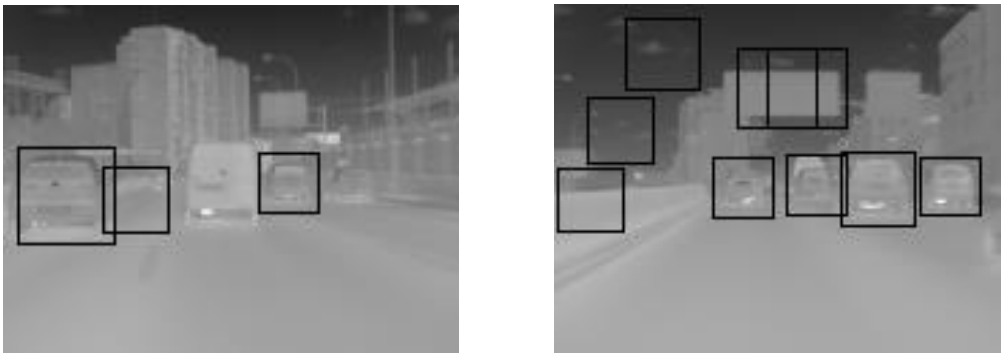


Figura 4.35: Winstride 8x8 píxeles

Subiendo a 16x16 píxeles se pierden muchos detalles y se dejan de detectar vehículos que previamente se reconocían perfectamente. Además, aquellas detecciones correctas, dejan de ser ajustadas a los vehículos, lo que implica un peor comportamiento.

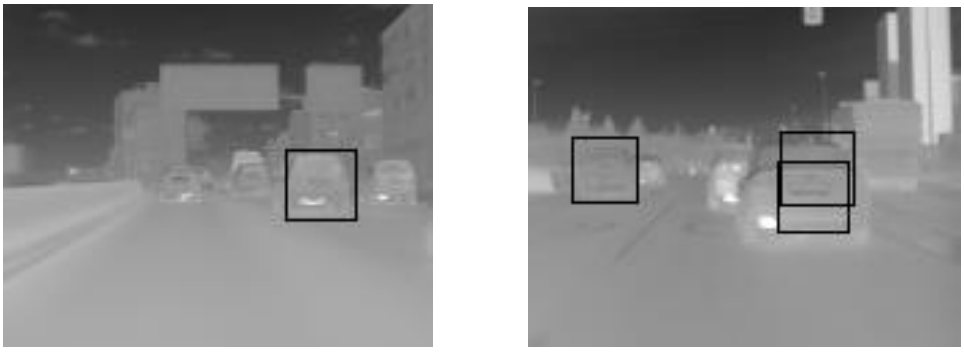


Figura 4.36: Winstride 16x16 píxeles

Esto lleva a la conclusión de que las mejores opciones son las dos primeras ventanas. En nuestro caso vamos a seguir con la ventana mediana, ya que oferta tiempos un poco más reducidos, aunque se podrían utilizar indistintamente cualquiera de las dos.

Padding

El padding toma algo más de importancia en este proceso, y en este caso, las detecciones se han podido realizar sin problemas, a pesar del tiempo que estas conllevaban. Partiendo de un entrenamiento donde esta variable tomaba un valor 0x0 pixeles, en la figura 4.37, podemos observar claramente cómo se pierde precisión en las detecciones, y sobre todo, que aquellos vehículos se encuentran en los laterales no son detectados. Esto se puede observar en las imágenes siguientes, las cuales reflejan los casos más comunes de este valor.

Figura 4.37: Padding 0x0 píxeles

Si este se aumenta a 4x4 píxeles, en la figura 4.38, esos pequeños defectos que se han mencionado son casi corregidos. Además, hay que tener en cuenta que aparecen más falsos positivos, pero estos son localizados en las zonas no aptas para la detección, por lo que no suponen un gran problema para el resultado final.

Figura 4.38: Padding 4x4 píxeles

Por último, se aumenta el valor del padding a 8x8 píxeles, pero este no ofrece resultados nuevos, como se aprecia en la figura 4.39. El tiempo de procesamiento es ligeramente superior, pero en cuestiones prácticas no es relevante. Es por ello que la solución final de esta variable para la búsqueda del detector óptimo se encuentra entre las dos últimas, utilizando nosotros la primera opción, 4x4 píxeles. Las siguientes imágenes corresponden al a última prueba realizada.

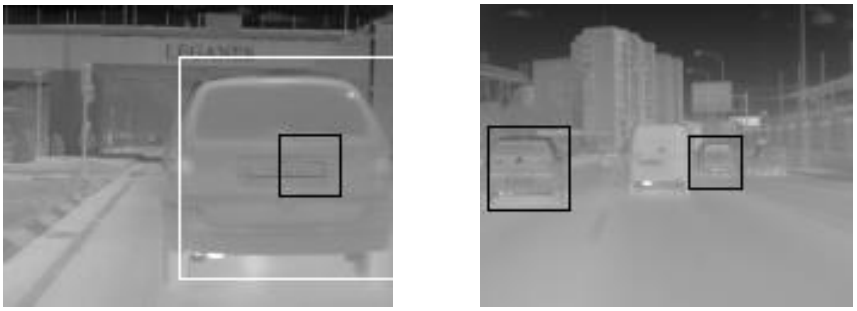


Figura 4.39: Padding 8x8 píxeles

Escala

Este parámetro, combinado con el tamaño de la ventana de desplazamiento, es que lo que nos proporciona la capacidad de conseguir mayor número de detecciones correctas, aunque esto también suponga mayor número de falsos positivos. Es uno de los más importantes, como se ha mencionado.

Es un parámetro que toma valores comprendidos entre 1 y 1.5, y sabemos, que cuanto menor sea este valor, mayor número de capas se verán involucradas. Tiene un gran impacto en el resultado final, pudiendo ofrecer desde un gran número de detecciones hasta ninguna. De esta forma, con un valor 1.01, como se puede observar en 4.40a, se consiguen numerosas detecciones, siendo estas además precisas. Como en casos anteriores, los falsos positivos que aparecen pueden ser controlados posteriormente, por lo que no es un gran problema.

(a) (b) (c)

Figura 4.40: (a) Escala 1.01 (b) Escala 1.3 (c) Escala 1.5

En las siguientes imágenes se observan los resultados para las escalas 1.3 y 1.5 y puede observarse claramente los resultados van empeorando conforme se aumenta el valore de esta variable.

El comportamiento medido en las escalas de precision y recall puede verse en la figura 4.41, la cual sigue el modelo de las ya explicadas.

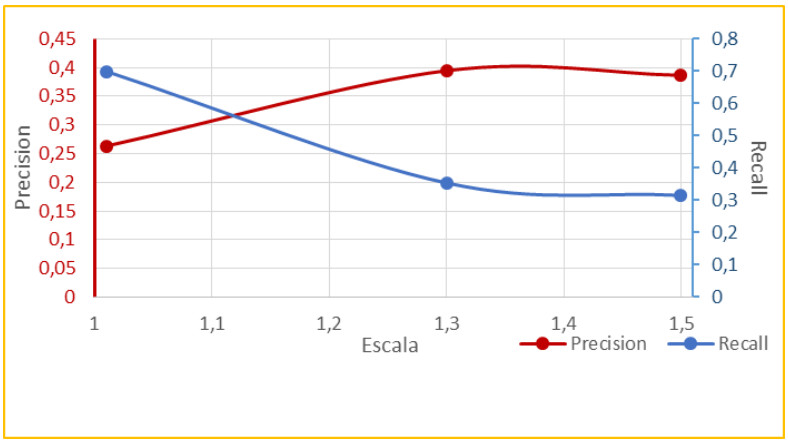


Figura 4.41: Precision - Recall Escala

Es por ello, que el mejor valor para este parámetro es el correspondiente a la primera prueba, donde se define como 1.01, ya que a pesar de tener una peor precision, el rango de detecciones verdaderas encontradas es mayor, y además, con los métodos de filtrado se podrán eliminar esos falsos positivos que emporan el primer parámetro.

Cabe mencionar que este parámetro es el principal causante de un mayor gasto computacional, y por lo tanto de un mayor tiempo de cómputo. En la siguiente gráfica, figura 4.42, se pueden observar estos hechos.

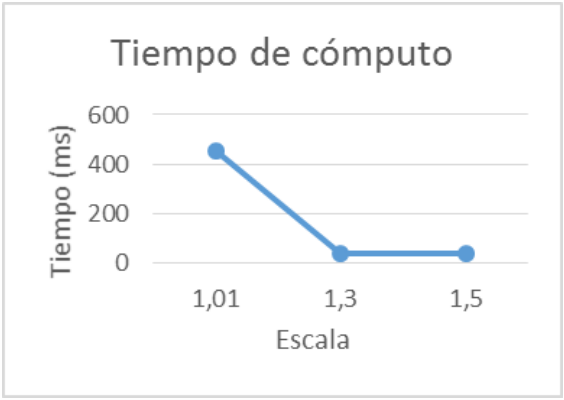


Figura 4.42: Tiempo de cómputo -Escala

GroupThreshold y Meanshift

Como se ha comentado, estos parámetros funcionan mejor conjuntamente. Estos se encargan de la eliminación de aquellas numerosas detecciones alrededor de un vehículo, lo cual indica que el vehículo está siendo visto desde varias posiciones. Para ver su funcionamiento se va a realizar una prueba con el parámetro de *meanshift* activo y no activo, para el mismo valor del *threshold*.

Así, en las figuras 4.43 y 4.44 podemos ver como con el *meanshift* desactivado, los

resultados de las detecciones no tienen gran cambio a los que se han ido mostrando anteriormente, mientras que una vez activado ese parámetro, los resultados cambian drásticamente, eliminando no solo aquellos grupos de detecciones alrededor de los vehículos, si no también algunas detecciones correctas.

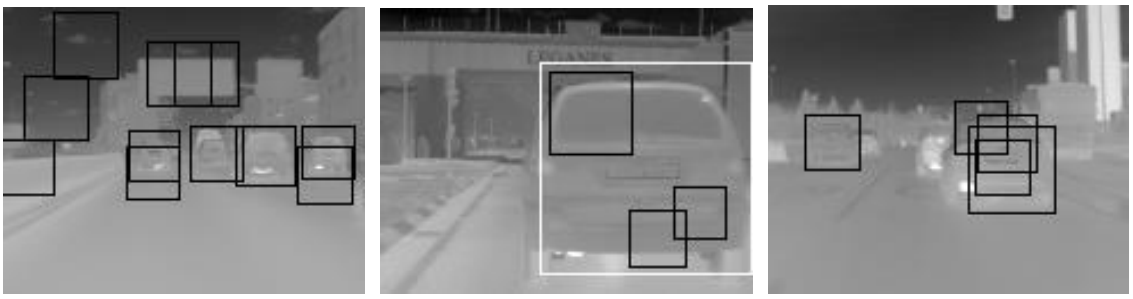


Figura 4.43: MeanShift desactivado

Figura 4.44: MeanShift activado

Es por ello que se decidió no utilizar este método para la eliminación de falsos positivos, si no el de *Non Maxima Supression* como se mostrará en el siguiente apartado, el cual ofrece mejores resultados al respecto.

La configuración final del proceso de detección quedará marcado por los siguientes valores:

$$\boldsymbol{hitThreshold}: \mathbf{0.05}$$
$$\boldsymbol{Winstride}: (\mathbf{8}, \mathbf{8})$$
$$\boldsymbol{Padding}: (\mathbf{4}, \mathbf{4})$$
$$\boldsymbol{Escala}: \mathbf{1.01}$$
$$\boldsymbol{GroupThreshold}: \mathbf{1}, \boldsymbol{false}$$

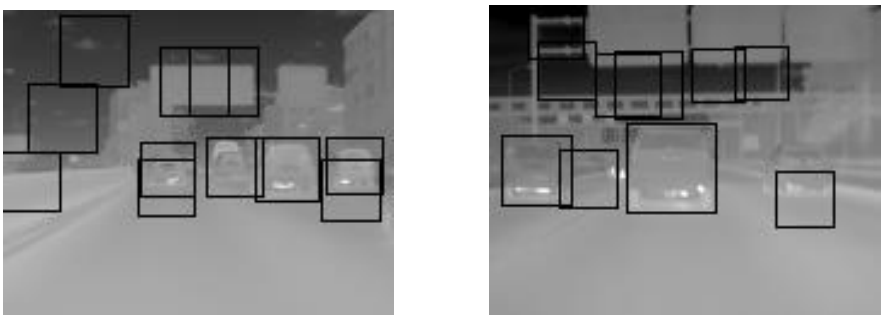
4.2.3.3 Otras características

En el caso del filtro de tamaño, más que un método de mejora, es un método de ayuda al a búsqueda. Puede observarse en las imágenes anteriores como existen rectángulos de dos colores. Los blancos son los detectados con el HOG sin aumentar la imagen, es decir, las detecciones que la propia función de *detectmultiscale* es capaz de percibir.

Son más exactos, y apenas generan falsos positivos, como se aprecia en la figura 4.45. Es por ello, que estas detecciones no son tratadas después por los métodos de filtrado posteriores. Los rectángulos negros son aquellos conseguidos a partir de aplicar la función de detección a las imágenes aumentadas. En esta vuelta sí se generan muchos falsos positivos que serán eliminados posteriormente.

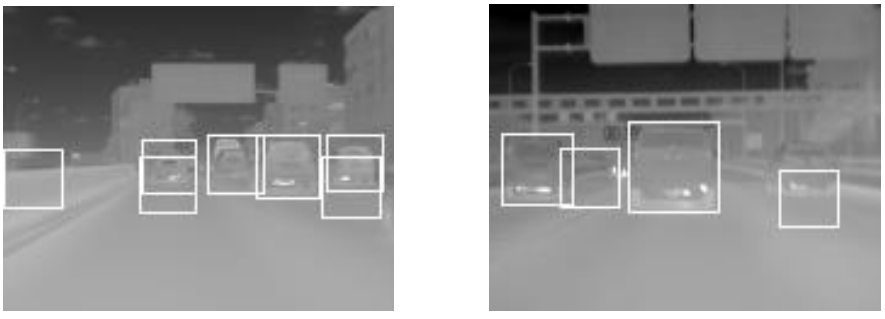
Figura 4.45: Resultado reescalado aumentado

El filtro de localización permite eliminar, en este caso, todas las falsas detecciones de las nubes, las farolas o los postes de información. Este es el que más falsos positivos ayuda a desechar, puesto que el detector tiende a confundirse bastante con esos objetos. Así por ejemplo, se pueden observar las diferencias entre la aplicación de este filtro y las detecciones originales, en las figuras 4.46 y 4.47.



(a) (b)

Figura 4.46: (a, b) Sin filtro de limitaciones



(a) (b)

Figura 4.47: (a, b) Con filtro de limitaciones

Por último el método del *Non Maxima Supression*, el cual no ha permitido eliminar aquellos falsos positivos detectados en la zona de trabajo. Este es regulado por un valor de umbral prefijado por el usuario, y aunque se ha aplicado a ambas detecciones, con la imagen en tamaño original y aumentándola, para el primer caso no es muy necesario puesto que las detecciones son bastante precisas, sin embargo, tenemos casos en los que se solapan, como puede verse en las imágenes de la figura 4.48. Los resultados de aplicar este algoritmo, se muestran en la 4.49.

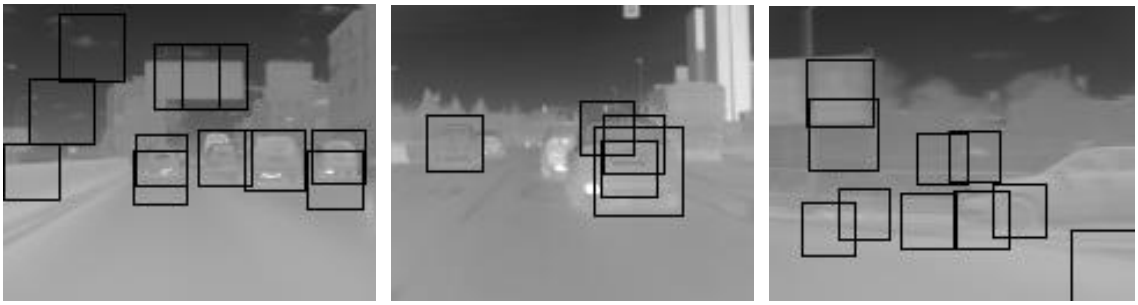


Figura 4.48: Sin NMS

Figura 4.49: Con NMS

El que durante la primera vuelta de detecciones, las obtenidas sean muy precisas, nos ha llevado a darle a estas algo de preferencia frente a las otras. Existían ocasiones donde dentro se encontraban detecciones dentro de detecciones, que no habían sido eliminadas por los filtrados explicados. Estas eran siempre las procedentes de la imagen aumentada, dentro de las detecciones originales sin aumentar la imagen. Es por ello que, además de eso filtros, se recurrió a la lectura de las posiciones de estos falsos positivos con respecto a los primeros detectados. Gracias a la función *contains (),* de la clase Rectángulo de C++, pudimos comprobar si un punto se encontraba dentro del área de ese rectángulo. De esta forma pudimos ser capaces de mejorar el detector y eliminar situaciones como las descritas, y como las recogidas en la figura 4.50. El resultado de la aplicación de este algoritmo se muestra en la imagen 4.51.

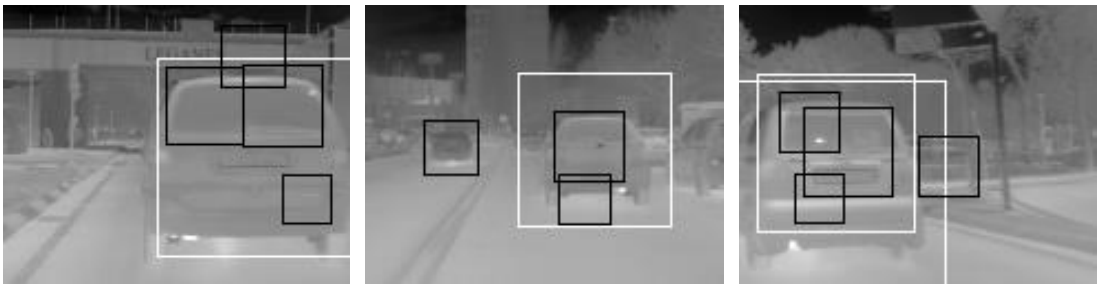


Figura 4.50: Sin NMS

Figura 4.51: Con NMS

El impacto que la aplicación de este código tuvo sobre las detecciones fue bastante relevante, ayudando a eliminar gran parte de aquellos falsos positivos existentes. Esto puede observarse en la gráfica 4.52, la cual recoge *precision* y *recall.*

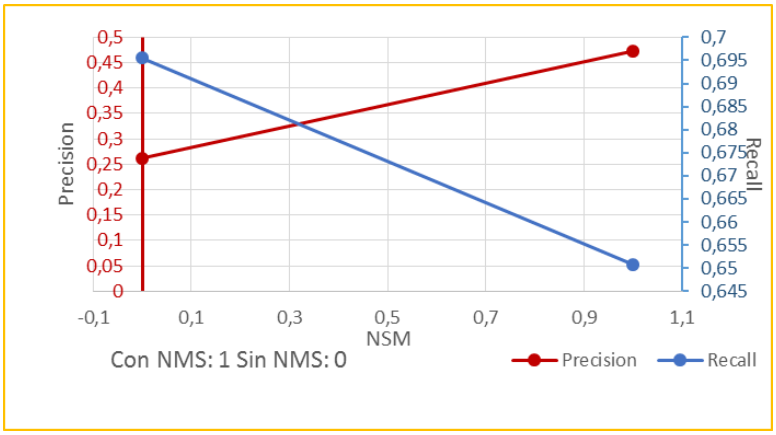


Figura 4.52: Precision-Recall NMS

4.2.4 Conclusiones del método

Tras los distintos análisis de la mayoría de los parámetros involucrados en este método, tanto en la parte del entrenamiento, como la de detección, se han podido sacar algunas conclusiones al respecto, que nos ayudaron a determinar el mejor comportamiento de los dos sistemas utilizados. Existen algunos puntos clave destacables de este proceso que se explicarán a continuación.

El primero de ellos está es la relación de imágenes utilizadas como base de datos del

entrenamiento. Tras la primera prueba con la relación 1:1, los entrenamientos se centraron en mejorar los resultados obtenidos por la relación 1:40. No fue hasta la modificación de gran número de parámetros cuando se observó que el sistema estaba siendo sobreentrenado de forma negativa, es decir, este estaba aprendiendo a reconocer que no era vehículo, en vez de que era vehículo. De esta forma, puesto que esa tarea era mucho más complicada, los resultados ofrecidos por esta relación estaban lejos de ser óptimos. Tras el cambio final del número de imágenes negativas, las detecciones empezaron a comportarse de una forma más coherente, dando lugar a los patrones de errores, o las zonas localizables.

Seguido de este anterior, cabe mencionar el tamaño de la imagen utilizada. Debido a que este método se basa en gradientes de la imagen, era de gran importancia la buena calidad de la imagen, así como su resolución o el tamaño de esta. Los resultados con el tamaño de 64x64 píxeles eran claramente mejores que los de tamaño inferior, por lo que, si el tamaño de la imagen hubiese sido mayor al presente, probablemente se hubiese podido trabajar con el tamaño de vehículo inferior, y no hubiese sido necesario el doble chequeo a una escala mayor.

Otro de los parámetros importantes de este método ha sido el ajuste de la variable del margen, c. Esta controlaba, de manera muy rigurosa, el comportamiento final del detector, como se ha podido observar, siendo decisivo su correcto valor para un comportamiento óptimo. Como se ha mencionado, si los resultados proporcionados por este parámetro no eran correctos, estos no podrían ser mejorados con otras variables, ni de entrenamiento ni de detección. La asignación de la unidad como valor de este dato, permitió, que junto con la correcta relación de imágenes, las detecciones cambiasen radicalmente, dando lugar a la solución que se ha ido mostrando a lo largo del análisis.

El cuarto parámetro a destacar es el de la escala del proceso de detección. En un primer momento, con los ajustes anteriores, las detecciones eran coherentes pero escasas, un detalle que frente a otros métodos de aprendizaje caracteriza a la combinación de HOG y SVM. Sin embargo, gracias a la escala, y por supuesto al ajuste de la búsqueda de objetos de dimensiones inferiores al tamaño del HOG, pudimos ampliar el número de verdaderos positivos, aunque esto implicase la aparición de falsos positivos también. El número de capas incluidas a la hora de la detección ha resultado ser francamente necesario para el resultado final obtenido.

Por último, el proceso que nos ha permitido eliminar esos falsos negativos que aparecían tras disminuir la escala ha sido la aplicación de la función NMS. En este caso, las limitaciones geográficas han sido determinantes solo para aquellas detecciones que se encontraban en el cielo, ya que no existían numerosos falsos positivos discordantes según su tamaño. Es por ello, que el gran problema estaba desencadenado por las detecciones en torno a un mismo objeto, lo cual se eliminó gracias a NMS.

Tras esto podemos concluir que el método de Dalal y Triggs [3], además de haber tenido una gran efectividad sobre peatones, es bastante positivo para la búsqueda de vehículos mediante cámaras infrarrojas. Los datos numéricos con las configuraciones finales de este método son los siguientes:

Verdaderos positivos: 548
Falsos positivos: 500
Falsos negativos: 204
Posibles detecciones: 752
Precision: 52%
Recall: 72.8%

Para el comportamiento obtenido, es un método de gran rapidez, si se compara con otros modelos de aprendizaje, tanto en la parte de entrenamiento como en la de detección. Es un método singularmente preciso, que podría tener un mejor comportamiento con una base de datos más grande, de la cual pudiese tener acceso a un considerable número de imágenes positivas, así como mayor diversidad de vehículos y modelos, ya que en los resultados pueden observarse ciertos modelos de vehículos que no terminan de ser reconocidos, aun así cuando claramente se aprecia un vehículo, mientras por la otra parte, el sistema reconoce vehículos que, por la borrosidad o la poca nitidez, no deberían ser reconocidos. Esto ha sido uno de los aspectos más destacables, y puede observarse en la figura 4.53.

Figura 4.53: Detalles particulares

4.3 Método 2: Haar Cascade

Como hemos podido observar, el método anterior tiene los dos procesos de entrenamientos claramente divididos, tanto en el desarrollo del método, como en su implementación práctica. En el caso del método de Viola Jones [2], no quedan tan definido, pero aun así permite una pequeña diferenciación entre ellos.

Para la implementación de este código únicamente se ha hecho uso de las librerías OpenCV, las cuales ya tienen incorporadas las funciones correspondientes tanto de la

creación de las características, como el entrenamiento del sistema. Esto, gracias a opencv_createsamples y opencv_traincascade es posible ejecutarlo desde la terminal, sin hacer uso de ningún entorno de trabajo, como el QtCreator. Sin embargo, también existe la posibilidad de su ejecución por esa vía.

La base de datos, tanto de positivos como de negativos es la misma, con el fin de poder comparar ambos métodos bajo las mismas condiciones. Sin embargo, la entrada de datos en el sistema, o en la función de generación de características, es distinta a la que vimos en el método anterior. En este caso es necesario crear dos ficheros de texto, uno para la carpeta de positivos, y otro para la carpeta de negativos:

- Carpeta de positivos: Es el archivo que ofrece la información real de los objetos, en el este caso de los coches. Este fichero debe recoger la ruta donde se encuentra la imagen, así como los objetos a detectar que hay en esta, indicando cuanto hay y cuáles son sus esquinas superiores izquierdas, así como cuáles son sus tamaños. Esto está pensado para imágenes no recortadas, imágenes en las que además de objetos hay otros elementos. En nuestro caso, para hacer uno de esos recortes del previo método, se recurrió a solo indicar la ruta y el tamaño de este. Al no indicar la posición del vehículo, el sistema entiende que se encuentra en toda la imagen. Así, nuestro fichero tendría la estructura indicada en la figura 4.54.

```
/home/carmen/Escritorio/imagenesBuenas/pos/imagen_resize0366.png        1 0 0 64 64
/home/carmen/Escritorio/imagenesBuenas/pos/imagen_resize0367.png        1 0 0 64 64
/home/carmen/Escritorio/imagenesBuenas/pos/imagen_resize0368.png        1 0 0 64 64
/home/carmen/Escritorio/imagenesBuenas/pos/imagen_resize0369.png        1 0 0 64 64
/home/carmen/Escritorio/imagenesBuenas/pos/imagen_resize0370.png        1 0 0 64 64
/home/carmen/Escritorio/imagenesBuenas/pos/imagen_resize0371.png        1 0 0 64 64
/home/carmen/Escritorio/imagenesBuenas/pos/imagen_resize0372.png        1 0 0 64 64
/home/carmen/Escritorio/imagenesBuenas/pos/imagen_resize0373.png        1 0 0 64 64
/home/carmen/Escritorio/imagenesBuenas/pos/imagen_resize0374.png        1 0 0 64 64
```

Figura 4.54: Fichero imágenes positivas

- Carpeta de negativos: en este caso, la generación del fichero es la misma, con la única diferencia que no hace falta indicar la posición de ningún elemento. Este archivo sólo recoge la ruta de cada imagen. Esto se observa en la figura 4.55.

```
/home/carmen/Escritorio/imagenesBuenas/neg/imagen_resize04017.png
/home/carmen/Escritorio/imagenesBuenas/neg/imagen_resize04018.png
/home/carmen/Escritorio/imagenesBuenas/neg/imagen_resize04019.png
/home/carmen/Escritorio/imagenesBuenas/neg/imagen_resize04020.png
/home/carmen/Escritorio/imagenesBuenas/neg/imagen_resize04021.png
/home/carmen/Escritorio/imagenesBuenas/neg/imagen_resize04022.png
/home/carmen/Escritorio/imagenesBuenas/neg/imagen_resize04023.png
/home/carmen/Escritorio/imagenesBuenas/neg/imagen_resize04024.png
```

Figura 4.55: Fichero imágenes negativas

4.3.1 Entrenamiento

Al igual que en el HOG+SVM, antes de poner el sistema a entrenar, es necesario recopilar la información del as características de las imágenes de muestra. Para ello se genera un vector con las características Haar que ya hemos comentado anteriormente. Esto se realiza con la función *opencv_createsamples*. Esta función, a partir de la base de datos proporcionada, crea nuevas muestras situando las imágenes positivas, aleatoriamente, por las imágenes negativas, siempre y cuando estas sean proporcionadas. Si no lo son, simplemente calcula las características de las muestras positivas de la base de datos. En el caso de hacerlo, crea más diversidad de posibles opciones. La salida de este proceso es un vector que recoge todas las características encontradas. Esta función cuenta con los siguientes parámetros:

- Vec: es el vector de salida. Este será posteriormente utilizado en la función opencv_traincascade con el fin de entrenar el sistema.

- Img o info: dependiendo de si la base de datos está compuesta por una o varias imágenes, con el fin de crear los ejemplos con la rotación aleatoria de estas, o si se trata de un conjunto, con la posibilidad de realizar la misma labor previa o simplemente extraer las características de este, se utiliza img o info respectivamente. La entrada de img es una imagen, con un tipo de archivo .jpg, .png o .jpeg, mientras que para el caso de info, la entrada es un fichero con las rutas de las imágenes, como se ha explicado anteriormente.

- Bg: se refiere a las imágenes de fondo o negativas, y es un parámetro opcional. La entrada de este parámetro es otro archivo de texto con la ruta de las imágenes.

- Num: recoge el número de muestras que queremos conseguir. Para los casos en los que no se quieran obtener nuevas muestras, si no las características de las proporcionadas, este parámetro tendrá el valor del número de imágenes positivas totales.

- Bgcolor: permite especificar el color de los fondos. Esto permite al sistema diferenciar mejor los objetos del fondo, tanto en las imágenes positivas proporcionadas, como en las imágenes de nueva creación, en el caso deseado. Por defecto esta en escala de grises, por lo que es un parámetro que no ha tenido que ser modificado.

- Maxxangle, maxyangle, maxzangle: para la generación de esas nuevas muestras posibles, además de situarlas aleatoriamente por los fondos de las imágenes negativas, también nos dan la posibilidad de añadir rotación en estas. Así, cada

parámetro anterior, indica las rotaciones en cada uno de los ejes. Ante una mayor heterogeneidad, mayor número de características relevantes, y por lo tanto mayor número de posibles detecciones de objetos cubiertas.

- w (width) y h (height): a pesar de ser un método no estricto con las dimensiones de las imágenes, como en el caso del HOG ya que todas tenían que ser iguales, para extraer las características de Haar, así como para entrenar, se nos pide unos parámetros de altura y anchura para establecer unas dimensiones de entrenamiento. Las imágenes tienen que ser de un tamaño parecido, pero no estrictamente el mismo.

Tras este proceso se obtiene el vector de características deseadas, y se podrá continuar con el entrenamiento. Para ello se recurre a la función de opencv_traincascade, el cual tiene los siguientes parámetros más relevantes:

- Data: es el archivo donde se guardará el sistema entrenado. Este se podrá recogerse de la carpeta de trabajo una vez terminado el entrenamiento.

- Vec: es el vector de características de las imágenes positivas extraídas.

- Bg: es el fichero de texto que recogía las rutas de las imágenes negativas. Esta vez sí es un parámetro obligatorio, puesto que para el entrenamiento es necesario la comparación con las negativas.

- numPos y numNeg: denominan el número de muestras, tanto positivas como negativas, que se utilizarán para cada clasificador, o nivel. El número de negativas es el número de imágenes que tenemos en nuestro directorio, mientras que el número de positivos es algo menor.

  Según lo explicado anteriormente, sabemos que entrenando el clasificador general, cuando una imagen es detectada como negativa en uno de los niveles, esta es eliminada de posibles opciones a clasificar. Teniendo en cuenta este detalle y sabiendo que numPos controla el número de muestras positivas sacadas en cada nivel, deducimos que es conveniente reducir el número de muestras exigidas a cada clasificador, frente al número de muestras que existen en el vector, ya que varias de ellas serán desechadas por el camino, y en el momento en el que el clasificador no pueda conseguir las mínimas muestras requeridas, este parará. De esta forma, si el número de positivas introducidas es 1000, será conveniente fijar este valor a 800, para evitar posibles errores durante en el entrenamiento.

- numStages: es el número de niveles de los que constará nuestro clasificador

final, en otras palabras, es el número de clasificadores fuertes existentes. Cuantos más niveles tengamos, mayor precisión conseguiremos en nuestras detecciones. Sin embargo, cada clasificador fuerte exige un gran coste computacional, por lo que hay que encontrar el equilibrio. Existen ocasiones en las que los intereses se consiguen antes de llegar al último nivel, por lo que el entrenamiento finaliza.

- Nsplits: hace referencia al a construcción de la cascada de clasificadores. Estos pueden estar dispuestos de forma lineal, uno tras otro, o en forma de árbol, lo que correspondería a una estructura con numerosas ramificaciones, y por lo tanto, caminos. Esto puede observarse en la figura 4.57.

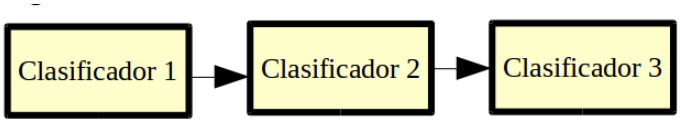


Figura 4.56: Clasificador lineal

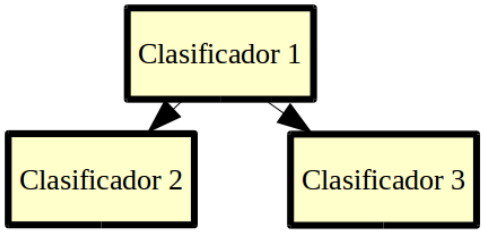


Figura 4.57: Clasificador en árbol

- featureType: se refiere al tipo de características con el que se trabaja al realizar los entrenamientos. Estas pueden ser de tipo Haar, como las obtenidas de las imágenes positivas o de tipo LBP, por sus siglas en inglés Local Binary Patterns. En cuanto a precisión, las características de Haar aportan mucha más información que las anteriores, sin embargo tienen un gasto computacional muchísimo mayor. Las LPB son más rápidas de calcular y con las que trabajar, pero no ofrecen los mismos resultados.

- w (width) y h (height): es, al igual que en la creación de las muestras, el tamaño deseado pero no estricto. Estas, o similares, serán las dimensiones de la ventana de detección.

- minHitRate: número mínimo de muestras positivas de cada nivel. Esta, junto con la siguiente son las condiciones de cada clasificador. Si no lo cumplen, los clasificadores simples de cada nivel son modificados para realizar una mejor tarea con los nuevos. El número mínimo de muestras global vendría designado por:

$$minHitRateGlobal = minHitRate^{number\_of\_stages} \quad (4.3)$$

- maxFalseAlarmRate: número máximo de muestras negativas por cada nivel. Si son superadas el clasificador fuerte cambia sus bases para buscar mejores resultados. En este caso, para calcular el valor global seria:

$$maxFalseAlarmRate = maxFalseAlarmRate^{number\_of\_stages} \quad (4.4)$$

Tras esto, obtenemos el clasificador en cascada que se utilizará para el detector. Este quedará recogido en un archivo .xml en la carpeta de trabajo.

4.3.2 Detección

La parte de la detección es similar en ambos métodos debido a que comparten numerosos conocimientos o bases, a excepción de la configuración de los clasificadores y las funciones a utilizar. Es por ello, que con el fin de optimizar los resultados, el código final incluye ambas opciones, con la posibilidad de utilizar una u otra según el usuario decida. De esta forma, la carga de imágenes, así como la muestra de las detecciones o el almacenamiento de estas, son común para ambas. Esto es debido a que se utilizan las mismas imágenes de testeo, y a que ambos detectores tienen como salida variables de rectángulos con la información de las detecciones.

4.3.2.1 Código utilizado

Configuración del clasificador Haar

La configuración del clasificador Haar ha sido muy similar a la realizada para el método previo. Las librerías OpenCV permiten la inclusión total del método de Viola Jones [2] gracias a las variables adaptadas para ello.

La configuración del modelo consiste en la lectura del fichero cascade.xml, que es el archivo resultante de los entrenamientos, y la asignación de la información que este ofrece a la variable correspondiente, la que nos permite trabajar con él.

Función de detección

Al igual que en el método anterior, toda la tarea de detección recae sobre la correcta configuración de los parámetros de una función, la cual utiliza el clasificador recién entrenado. Esta función, bajo el mismo nombre que la del HOG, *detectmultiscale*, aplica los conocimientos mencionados de Haar, con el fin de mejorar las detecciones originarias de los entrenamientos. Así pasaremos por cada parámetro de esta función, la

cual nos permitirá trabajar los resultados finales.

- Imagen: es la imagen de fuente que se utiliza para captar las detenciones. Vuelve a ser el único parámetro obligatorio junto con el vector de rectángulos detectados, Rect. De igual forma al *detectmultiscale* del HOG, la imagen puede ser introducida en color o en escala de grises, y en cualquier tamaño.

- Rect: es la variable rectángulo donde se guardan las detecciones de la imagen, las coordenadas de las esquinas importantes y el valor del ancho, y alto, de este.

- scaleFactor: el valor de la escala funciona de igual forma que el método anterior. Este controla el número de capas a realizar durante el reescalado, interno, de la función. Cuanto menor sea la reducción de la imagen, más capas obtenemos, y por lo tanto, más posibilidades de encontrar vehículos. El valor predeterminado suele estar alrededor de 1.05. Este valor significa que cada vez que se reduzca la imagen, lo hará un 5%, por lo que el número de capas será mayor que si se reduce, por ejemplo, un 50%, y por lo tanto existe más probabilidades de encontrar un objeto.

- MinNeighbors: es un parámetro que podría considerarse equivalente a lo que *meanshift* era en la función del HOG. Se encarga de contar cuantas detecciones hay para la misma zona de la imagen, y según un valor predefinido por el usuario, mostrar únicamente una única detección.

  Imaginémonos una detección de vehículos, y alrededor del objeto en cuestión nos encontramos con 4 rectángulos de detección. Si el parámetro de MinNeighbors tiene un valor 3, esas varias detecciones pasarán a convertirse en sólo una. Sin embargo, si en vez de tomar el valor 3, le damos el valor 5, al mostrar las detecciones, no se dibujará ningún rectángulo, puesto que no han pasado el umbral límite.

  Esto significa que cuanto mayor este valor sea, más preciso será el resultado final, pero hay que saber encontrar el valor correcto. Es utilizado junto al parámetro de la escala para encontrar el mayor número de objetos, y con este último, eliminar todos aquellos que sean falsos positivos.

- Flags: este valor no suele ser usado en las nuevas cascadas, puesto que fue desarrollado para los primeros modelos. Controla cómo eliminar algunas zonas que contienen muchas o pocas detecciones mediante el sistema de Pruning de Canny. Actualmente esto es mejorado con el parámetro anterior, así como por la evolución de los entrenamientos.

- MinSize: el tamaño mínimo que se busca en la imagen para las detecciones, siendo

este superior al tamaño del a ventana de entrenamiento. Al igual que el método anterior, este no es capaz de detectar objetos inferiores al tamaño de entrenamiento.

- MaxSize: el tamaño máximo de detección. Este no suele ser utilizado, a no ser que la aplicación este centrada en un sector concreto de tamaños. En nuestro caso será utilizada para evitar que el sistema alargue la búsqueda para unas dimensiones mayores de lo que nuestros vehículos pueden medir.

4.3.3 Resultados

4.3.3.1 Análisis de los parámetros de entrenamiento

Los entrenamientos de las cascadas de Haar hacen uso de un mayor tiempo de cómputo debido al gran número de características con las que trabaja. Esto, en gran medida, ha dificultado el análisis de los distintos parámetros, puesto que no ha sido posible realizar un entrenamiento para cada pequeño cambio. De esta forma, se ha conseguido llegar a una configuración aceptable, mediante la modificación de tan sólo algunos parámetros, lo que, es posible, que se pueda mejorar su rendimiento teniendo en cuenta todas las variables a modificar.

Para el entrenamiento se utilizan las funciones *opencv_traincascade* y *opencv_createsamples*. Esta última función no tiene parámetros relevantes que mencionar, puesto que sólo se encargaba de crear el vector de características Haar y lo hemos basado en las imágenes de entrada, sin crear nuevas negativas mediante rotaciones de las imágenes positivas. Es por ello que a continuación se comentarán, exclusivamente, los parámetros de la primera función, así como algunos aspectos a tener en cuenta de las imágenes de entrada.

Tipo de imagen

Con el fin de aprovechar los recortes utilizados por HOG, que recordamos que era la imagen del vehículo más un pequeño margen que nos permitía realizar correctamente el histograma de gradientes, se utilizaron esas figuras como la base de datos de este nuevo método. Sin embargo, tras numerosas pruebas y cambios de parámetros, se observó, cómo las detecciones no terminaban de ser concisas, lo que daba lugar a un mal comportamiento del detector, tal y como se observa en la figura 4.58.

Figura 4.58: Resultados con base de datos con margen

Es por ello que se modificó la base de datos del entrenamiento, y esta vez se introdujeron los recortes de los vehículos, sin ese margen que se añadía para HOG. Este daba lugar a confusiones a la hora de calcular las características de Haar, y por lo tanto, no realizaba correctamente el trabajo de clasificación. Una vez cambiadas las imágenes, las detecciones comenzaron a ser más fiables, obteniendo mejores resultados que las anteriores.

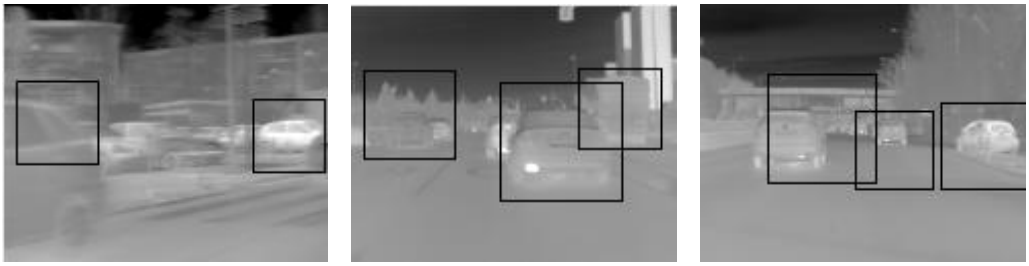


Figura 4.59: Resultados con base de datos sin margen

Como se puede apreciar en las figura 4.59, las detecciones aumentan considerablemente, incluyendo, a su vez, un mayor número de falsos positivos. Sin embargo además, aumenta el número de verdaderos positivos, y disminuye el de falsos negativos, lo cual muestra claramente la mejora del comportamiento del clasificador, ya que los falsos positivos, pueden ser eliminados posteriormente. Es por ello que como base de datos se eligió la de los recortes de vehículos ceñidos a los bordes del objeto, como positivas, y las mismas negativas que para el método anterior.

Estos resultados se pueden observar en la gráfica de la figura 4.60, la cual nos muestra como el comportamiento general del detector ha mejorado considerablemente al cambiar el tipo de imagen.

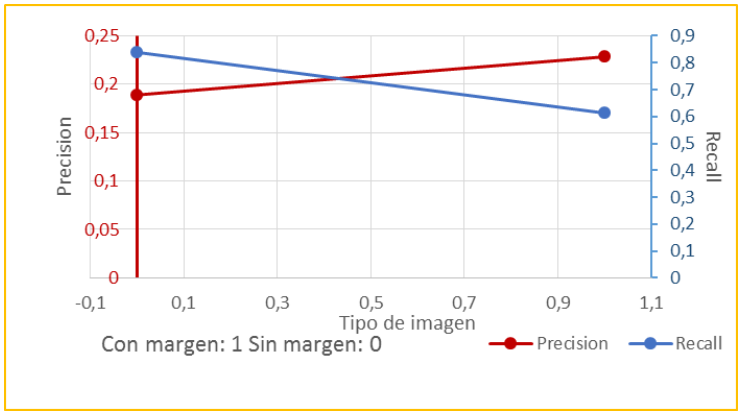


Figura 4.60. Precision - Recall Tipo de imagen

Tamaño de la imagen

Con una relación base de imágenes positivas y negativas, respectivamente, de 1:2.5, se calcularon las cascadas para los distintos tamaños a estudiar, con el fin de analizar cuál era el mejor tamaño de imagen para la obtención de los deseables resultados.

Empezando por las imágenes de mayor tamaño, puestos que estas ofrecerían un mayor número de características, y por lo tanto, mayor exactitud, se seleccionó un tamaño de 64x64 píxeles. Esto dio lugar a entrenamientos largos, entre 48 y 72 horas, debido al trabajo con un mayor número de distintivos de cada imagen. Sin embargo, los resultados obtenidos no eran los deseados. El tamaño máximo de detección era el del entrenamiento, 64x64 píxeles, y para la búsqueda de elementos más pequeños, si se aplicaba el filtro de tamaño que se comentó anteriormente, este, además de necesitar un tiempo de cómputo muy elevado, no encontraba suficientes objetos. Este comportamiento puede observarse en la figura 4.61, donde los vehículos a distancias mayores no son detectados.

Figura 4.61: Tamaño de entrenamiento 64x64 píxeles

Con el fin de mejorar lo recién obtenido se disminuyó el tamaño de la imagen de entrenamiento, permitiendo así al clasificador buscar objetos más reducidos sin la necesidad de aumentar la imagen. Según podemos comprobar en la figura 4.62, la solución en este caso fue ligeramente mejor, aunque lejos todavía de tener un comportamiento óptimo. Las detecciones intentaban buscar lo vehículos, pero no terminaban de encasillarlos correctamente.

Figura 4.62: Tamaño de entrenamiento 48x48 píxeles

Por último, se realizó el entrenamiento con imágenes de 32x32 píxeles. Al disminuir el tamaño de la ventana de entrenamiento, se optaba a la detección de objetos de dimensiones inferiores. Los resultados quedan recogidos en la figura 4.63.

Figura 4.63: Tamaño de entrenamiento 32x32 píxeles

En este caso vemos una clara mejora. Existen numerosas detecciones, notablemente más enfocadas en el objeto, a diferencia de los anteriores. Podemos observar como el número de falsos positivos ha aumentado considerablemente, pero, dado que los falsos negativos han disminuido, podemos considerar este tamaño de imagen de entrada el correcto para futuras pruebas. Contamos con unos datos numéricos de 516 verdaderos positivos, frente a los posibles 752 detecciones de coches, y un número de falsos positivos de 4778. Estos últimos intentarán ser eliminados, tanto en los entrenamientos como en el procedimiento de detección, gracias a los distintos parámetros de las funciones a utilizar.

En la figura 4.64 se observan los efectos de cada tamaño en las detecciones finales, observando como el tamaño de 32 píxeles es el que mayor *recall* ofrece, aunque la precisión no sea elevada. Estos falsos positivos podrán ser eliminados posteriormente, ya que son fácilmente localizables.

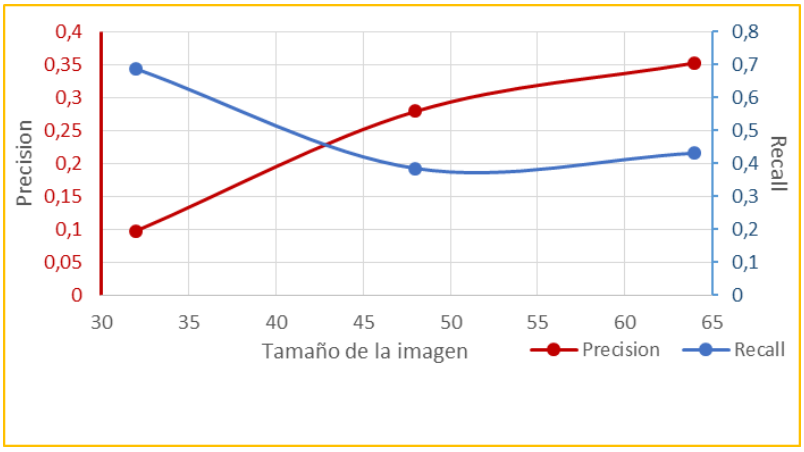


Figura 4.64: Precision - Recall Tamaño de la imagen

Número de imágenes

El siguiente paso fue la determinación de la relación correcta de positivas y negativas, puesto que aunque las detecciones en las pruebas anteriores fuesen medianamente aceptables, estas se podrían mejorar gracias al aumento de imágenes negativas, proporcionándole al sistema un mayor número de objetos que podrían evitar futuras

confusiones en las detecciones. Sin embargo, esto traía añadido un problema, ya que a mayor número de imágenes de trabajo, mayor es el tiempo de cómputo. Es por ello que habría que encontrar la relación de equilibrio entre mejores resultados y tiempo, o gasto computacional, requerido, como ya ocurrió en casos anteriores. Además, si el número de imágenes negativas se aumenta demasiado, el sistema podría reaccionar de forma negativa, y en vez de aprender qué es coche, su comportamiento se correspondería a la búsqueda de "no negativos", dando lugar a peores resultados finales.

Para una relación de imágenes 1:2.5, los resultados eran sensiblemente favorables, como se puede observar en la figura 4.65.

Figura 4.65: Relación imágenes 1:2.5

Aumentando la relación a 1:5, empleando así unas 8000 imágenes negativas, según las imágenes de la figura 4.66, obtenemos unos resultados más coherentes. Estos consiguen un mayor número de detecciones correctas, y aunque los falsos positivos vuelvan a verse aumentados, estos podrán ser eliminados, o al menos reducidos, posteriormente con algunas herramientas.

Figura 4.66: Relación imágenes 1:5

Se aumentó una vez más el número de negativas a 16000, con el fin de observar si seguían existiendo mejoras en el sistema. Sin embargo, esto no fue así, dando lugar a resultados tenuemente más deficientes que la prueba anterior, como se observa en la imagen 4.67.

Tras este análisis se eligió la relación de 1:5, siendo esto 1600 positivas y 8000 negativas, como relación de imágenes de la base de datos del clasificador final.

Figura 4.67: Relación imágenes 1:10

NumStages

El número de pasos, o de cascadas débiles, con los que contaríamos en la cascada fuerte, venia designada por este parámetro. En los primeros entrenamientos realizados, aquellos en los cuales las imágenes positivas introducidas contaban con un margen añadido, la cantidad de niveles alcanzada en los entrenamientos no era mayor de 7 u 8. Esto estaba ocasionado por las imágenes incluidas, ya que no le ofrecían más información, y por lo tanto, el clasificador final no podía cumplir con los requisitos mínimos.

Una vez modificada esta cuestión, los procesos se extendían a más niveles, llegando a alcanzar los 18 pasos. Con cada nueva cascada se mejoraban notablemente los resultados, pero tenía el gran inconveniente de exigir un gasto computacional desmesurado, llegando a superar las 60 horas de entrenamiento.

En este caso, el valor óptimo para este parámetro fue el de 17 niveles, ya que para el tiempo empleado, con respecto a clasificadores más precisos, obtenía resultados ampliamente concisos. Las diferencias entre detectores con distintos niveles, pueden apreciarse en las figuras 4.68, con 13 *stages*, y 4.69, con 17 *stages*, observándose como la exactitud es conseguida con el segundo caso.

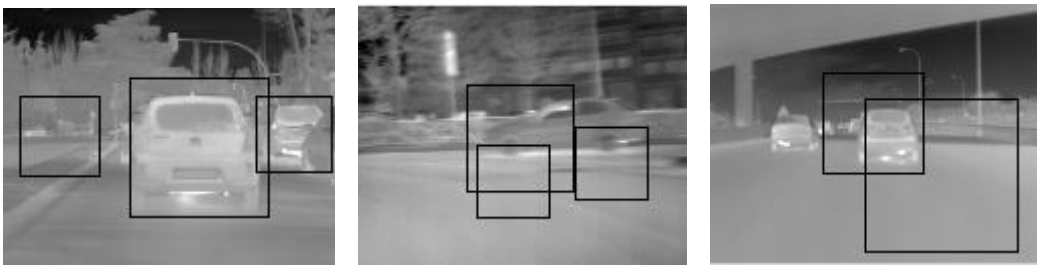


Figura 4.68: NumStages 13

Figura 4.69: NumStages 17

Nsplits

Este parámetro, el cual marca la disposición de los clasificadores débiles, puede ser configurado como árbol, introduciendo un número superior a 0, indicando este el número de ramificaciones, o 0, asemejándolo a un comportamiento lineal.

Figura 4.70: Nsplits 0

En la figura 4.70 podemos ver el resultado de este último caso. Vemos como las detecciones son óptimas, aunque existan algunos falsos positivos. Por otro lado, en la figura 4.71 nos encontramos con los resultados de un clasificador con estructura árbol y cinco ramificaciones. Podemos observar como los resultados son singularmente similares, a excepción de algunos falsos positivos incluidos en el segundo caso. Es por ello, que de cara a la configuración final se selección un clasificador de estructura simple, sin ramificaciones existentes.

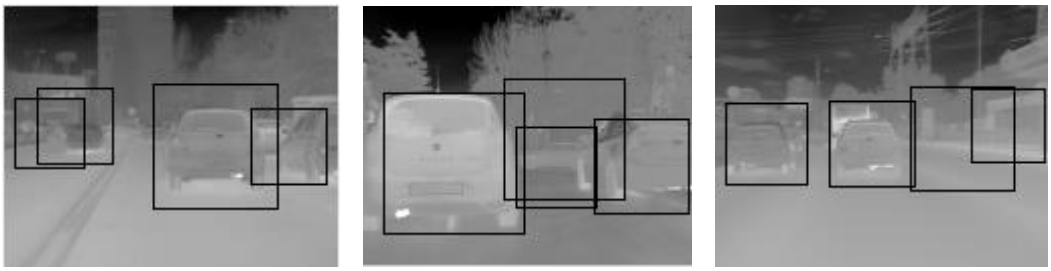


Figura 4.71: Nsplits 5

FeatureType

El tipo de característica es la base de extracción de la información de las imágenes de la base de datos. Al realizarlo mediante el procedimiento de Haar, el tiempo de procesamiento aumenta ligeramente, pero los resultados que se obtienen suelen ser más favorables.

Sin embargo, estas características también pueden ser extraídas mediante LBP, lo cual hace el proceso más rápido, puesto que la ejecución y el cálculo de estas no requiere tanta duración, pero, a pesar de seguir una línea de resultados muy similar a la de Haar, este valor pierde precisión en las detecciones, no tanto en la inclusión de falsos positivos, si no en la de falsos negativos, y por lo tanto, la pérdida de detecciones

correctas. Esto puede observarse en las figuras 4.72 y 4.73.

Es por ello que se eligió las características Haar para el clasificador final.

Figura 4.72: Tipo Haar

Figura 4.73: Tipo LBP

MinHitRate

Una condición de los clasificadores débiles, la cual deben cumplir con el fin de seguir con el proceso. Cuanto menor sea el número de imágenes positivas exigidas a cada clasificador, más sencillo será obtenerlas y por lo tanto, pasar al siguiente nivel, lo que significa que la precisión, a su vez, también disminuirá.

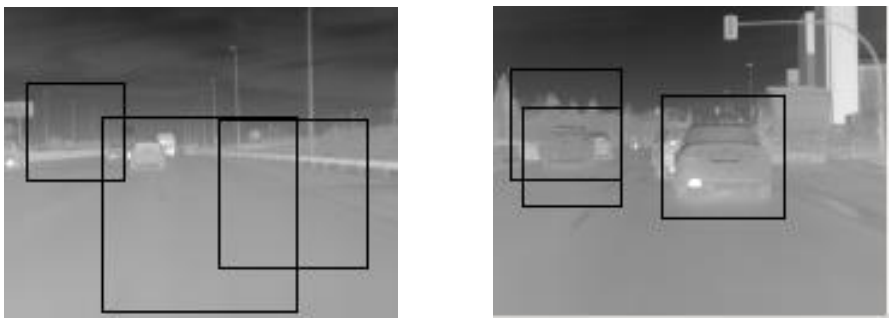


Figura 4.74: MinHitRate 0.2

Esto puede observarse en las figuras 4.74 y 4.75, las cuales muestran las diferencias entre un entrenamiento con el parámetro ajustado a 0.2 y a 0.99, respectivamente.

Figura 4.75: MinHitRate 0.99

Tras el análisis de los resultados, el valor óptimo para este parámetro es el equivalente a 0.99, ya que es el que indica una mayor precisión, al exigir un número mínimo elevado de detecciones positivas.

MaxFalseAlarmRate

Junto con el anterior, es una de las condiciones de cada clasificador débil. Al igual que previamente, cuando menor sea la exigencia, menor será el tiempo de cómputo y mayor será el número de errores en las detecciones. Esto puede observarse en las siguientes figuras. Es por ello, que el valor elegido es 0.5, ya que ofrece mejores resultados. El número de falsos positivos se ve reducido, y las detecciones son más exactas, como se puede comprobar en las figuras 4.76 y 4.77, recogiendo estas el comportamiento para un valor de 0.2 y 0.5 respectivamente.

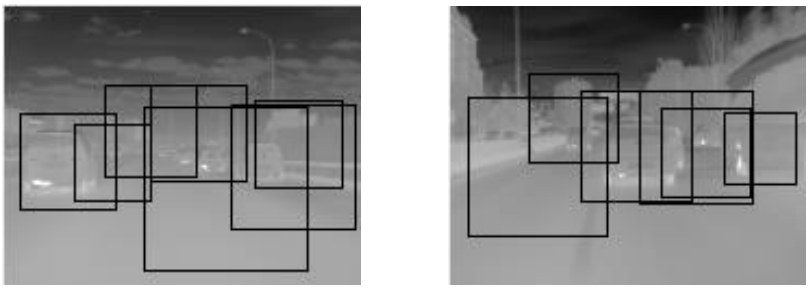


Figura 4.76: MaxFalseAlarmRate 0.2

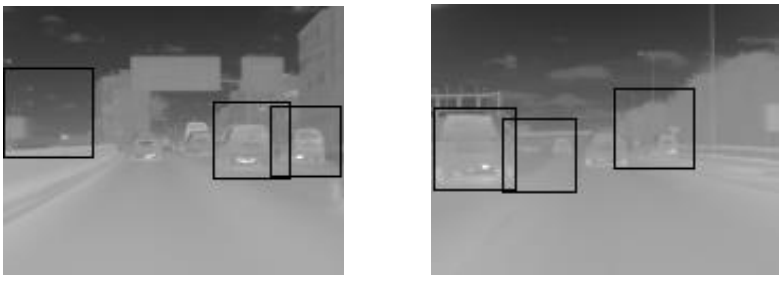


Figura 4.77: MaxFalseAlarmRate 0.5

Tras los distintos análisis y modificaciones, llegamos a la conclusión que la configuración de entrenamiento que nos ofrece mejores resultados es la que viene recogida a continuación:

$$\boldsymbol{Tipo\ de\ imagen}: \textbf{sin}\ \boldsymbol{margen}$$
$$\boldsymbol{Tamaño\ imagen}: (\mathbf{32}, \mathbf{32})$$
$$\boldsymbol{Numero\ de\ imagenes}: \mathbf{1}: \mathbf{5}$$
$$\boldsymbol{NumStages}: \mathbf{17}$$
$$\boldsymbol{Nsplits}: \mathbf{0}$$
$$\boldsymbol{FeatureType}: \boldsymbol{Haar}$$
$$\boldsymbol{MinHitRate}: \mathbf{0.999}$$
$$\boldsymbol{MaxFalseAlarm}: \mathbf{0.5}$$

4.3.3.2 Análisis de los parámetros de detección

ScaleFactor

El factor de la escala, que como se ha comentado, reacciona de igual manera que el parámetro escala de HOG. Cuanto menor sea su valor, siempre en un rango de 1 y 1.5, respondiendo eso a 100% de la imagen o el 150%, mayor será el número de capas, y por lo tanto el número de detecciones. Se puede definir como el paso de ampliación.

En la figura 4.78, podemos observar las respuestas para las escalas 1.001, 1.025 y 1.5 respectivamente. Observamos como en el primer caso aparecen un mayor número de falsos positivos, mientras que en los otros dos estos desaparecen. De hecho, en el último ejemplo podemos observar como la detección correcta del vehículo no es encontrada con esta escala. Es por ello, que frente a los resultados obtenidos, el valor de la escala será 1.025, ya que encuentra el equilibrio entre los falsos positivos y negativos.

(a) (b) (c)

Figura 4.78: (a) Escala 1.001 (b) Escala 1.025 (c) Escala 1.5

Se ha recogido el comportamiento de esta en un gráfico, el cual corresponde con la figura 4.79, con el fin de comprar *precision* y *recall.* Se puede observar como el comportamiento es muy similar al que el parámetro "escala" tenía en el método de HOG. Cuando menor es el parámetro, menor es el paso de un nivel a otro, y por lo tanto aumenta el número de capas de detección, lo que consecuentemente, aumenta el número de detecciones. Además, el tiempo de cómputo necesario también actúa de igual

manera. En la figura 4.80, se puede apreciar como al aumentar el número de capas, este también se ve afectado proporcionalmente.

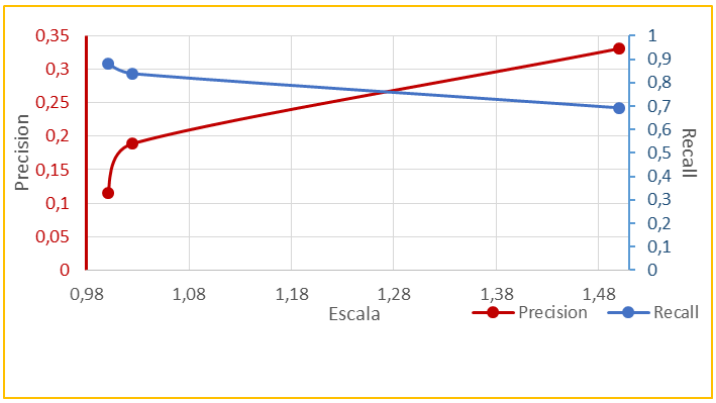


Figura 4.79: Precision- Recall ScaleFactor

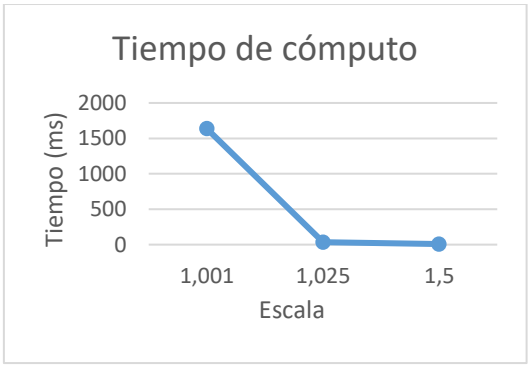
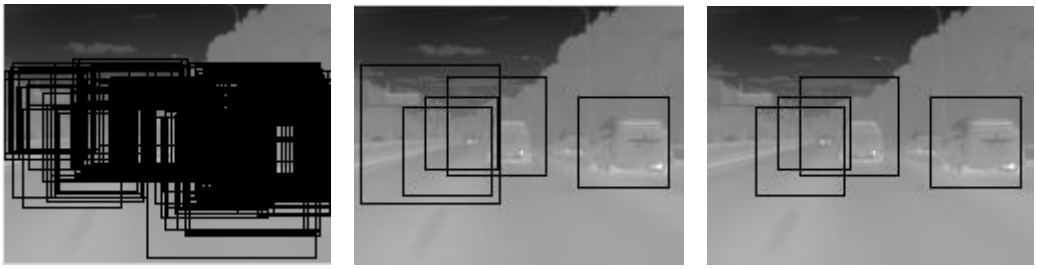


Figura 4.80: Tiempo de cómputo

MinNeighbors

Este evita las numerosas detecciones en la zona de un mismo objeto, como se puede observar en la figura 4.81a.



(a) (b) (c)

Figura 4.81: (a) MinNeighbors 0 (b) MinNeighbors 1 (c) MinNeighbors 5

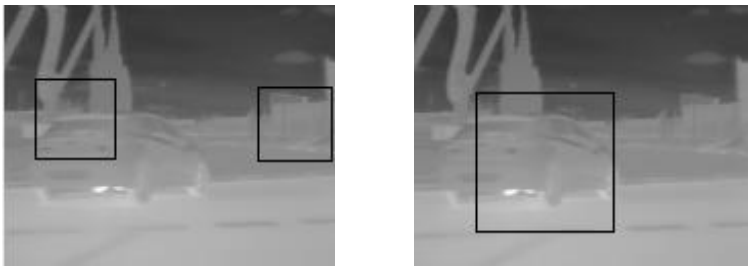
Controlando el numero de rectangulos detectados en las cercanias de los respectivos, estos pueden ser eliminados facilmente. Es por ello, que conforme se va aumentando el numero minimo de rectangulos vecinos que debemos cumplir, como se puede observar en la figuras 4.81b y4.81c, las detecciones se van haciendo mas exactas. Sin embargo,

esto tambien acarrea un problema, el cual es la eliminacion de detecciones correctas, como se ha comprobado en otras secuencias. Debido a este motivo, se he preferido elegir la unidad como valor de este parametro, e intentar eliminar esos falsos positivos mediante otras herramientas.

MinSize y MaxSize

El limite, tanto superior como inferior, de nuestra búsqueda dará lugar a la eliminación de aquellas detecciones que no cumplen con los requisitos, pero además, supondrá un cambio en la búsqueda de objetos, puesto que los resultados pueden ser favorables o desfavorables.

En el caso de las figuras 4.82a y 4.82b, podemos observar como el resultado ha mejorado al limitar el mínimo tamaño, ya se ha conseguido una mayor, y mejor, detección, mientas se ha eliminado un falso positivo. Sin embargo, este comportamiento depende de la imagen a estudiar.



(a) (b)

Figura 4.82: (a) MinSize (32, 32) (b) MinSize (40, 40)

Mientras tanto, en el caso de las figuras 4.83a y 4.83b, observamos como de una detección prácticamente aceptable, hemos pasado a dos detecciones de vehículos más pequeñas, debido a la limitación superior, y que por lo tanto no engloban correctamente los objetos. Es por ello, que para esta aplicación, se ha preferido no limitar las detecciones mediante este método, ya que posteriormente, se utilizara el filtro por locación, ya explicado anteriormente.

(a) (b)

Figura 4.83: (a) MaxSize (200,200) (b) MaxSize (50, 50)

Tras esto podemos concluir que la configuración de la detección es la que se recoge a continuación:

$$\boldsymbol{scaleFactor}: \mathbf{1.025}$$
$$\boldsymbol{MinNeighbors}: \mathbf{1}$$
$$\boldsymbol{MinSize}: --$$
$$\boldsymbol{MaxSize}: --$$

4.3.3.3 Otras características

De igual forma que para el método anterior, en el proceso de detección se han utilizado los algoritmos anteriores de filtrado de falsos positivos. Así, gracias al limitador por localización, se han conseguido eliminar numerosas detecciones erróneas, ya que el principal problema de Haar Cascades es la cantidad de falsos positivos que incluye en la imagen. Esto puede observase en las figuras 4.84 y 4.85, los cuales, respectivamente, corresponden a respuestas antes y después de aplicar el filtro de localizaciones.

Figura 4.84: Sin limitaciones geográficas

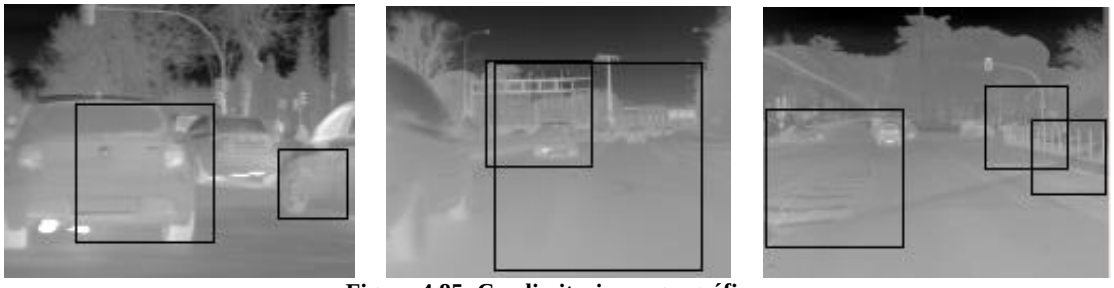


Figura 4.85: Con limitaciones geográficas

Es uno de los procedimientos que más efecto ha tenido en la mejora de las detecciones, gracias a la eliminación de esos numerosos falsos positivos existentes. El número de detecciones erróneas previas a la aplicación de este filtro era de 2708, mientras que tras su aplicación, este se vio reducido a más de la mitad, siendo 1102 detecciones. Esto, en consecuencia, afecta a la *precision* y *recall*, como se observa en la figura 4.86.

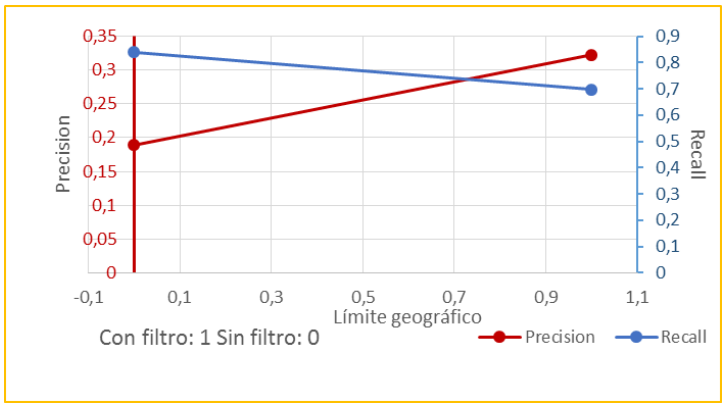


Figura 4.86: Precision-Recall Límite Geográfico

De igual forma, se ha hecho uso de la función de NMS para eliminar aquellas detecciones que intentaban definir el mismo objeto, que todavía podrían existir tras la asignación de la unidad a MinNeighbors. Así, para esta ocasión, como las detecciones en Haar no son tan ceñidas al objeto, al contrario que HOG, se ha aumentado el nivel del umbral de solapamiento.

Así, los resultados de las detecciones sin aplicar el NMS se podrían ver reflejados en la figura 4.87. Podemos observar como existen numerosos solapamientos, los cuales dificultan el resultado final de la imagen. Tras aplicar el método en cuestión, según la figura 4.88, estos se corrigen notablemente.

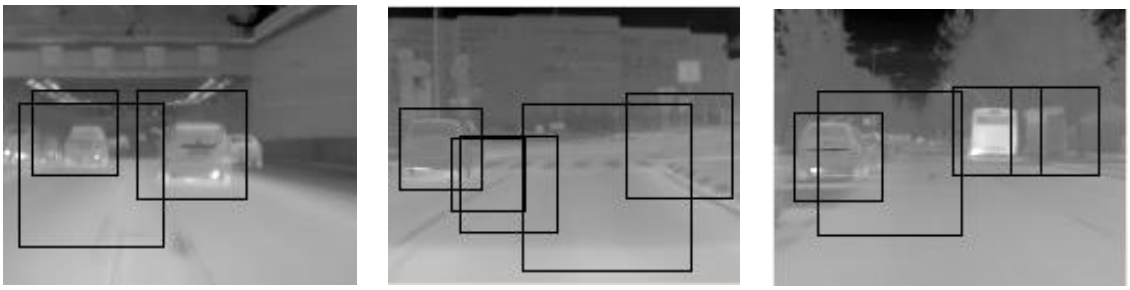


Figura 4.87: Sin NMS

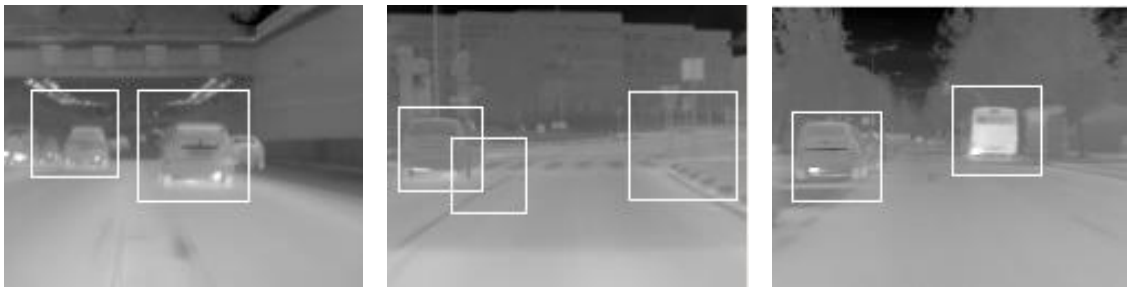


Figura 4.88: Con NMS

Para mostrar el impacto que puede tener el umbral de solapamiento en las detecciones, se ha aumentado su valor para la última secuencia. Podemos observar como los rectángulos sobrantes desaparecen, pero a su vez, hemos eliminado la detección

verdadera del coche de la derecha. Este es uno de los grandes inconvenientes de esta función, la cual no siempre tiene un comportamiento deseado.

Figura 4.89: Con NMS y mayor margen

Este, junto con el anterior proceso, nos ha permitido eliminar los falsos positivos. En este caso, la influencia que el NMS ha tenido en el resultado final es la recogida en la gráfica de la figura 4.90.

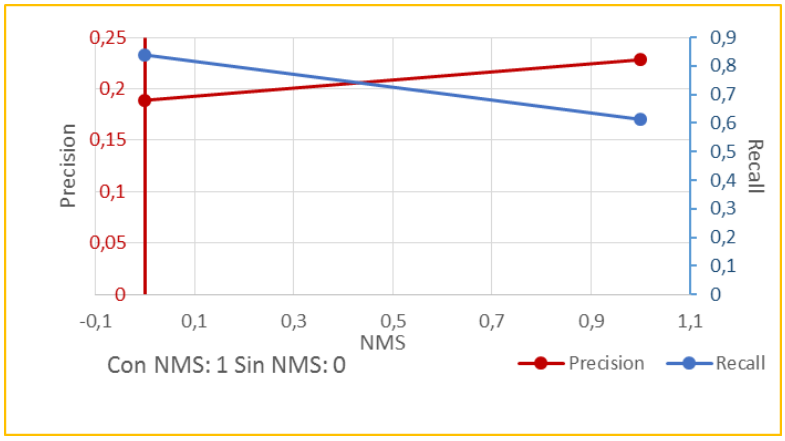


Figura 4.90: Precision- Recall NMS

Como se ha comentado, estas funciones no siempre actúan de forma correcta, y como se puede observar, al aplicar el filtro, el parámetro *recall* se ve disminuido en un 20%, lo que implica que se pierden detecciones correctas. Sin embargo, el número de falsos positivos también disminuye, lo cual es el objetivo de estos procesos.

La combinación de estos dos, limitaciones geográficas y NMS, es lo que nos ha permitido mejorar notablemente los resultados, reduciendo, sobre todo, las detecciones erróneas, o en otras palabras, los falsos positivos.

4.3.4 Conclusiones del método

Haciendo balance de todo lo estudiado, y observado, para Haar, es conveniente afirmar que el detalle esencial que permitió pasar de detecciones pobres a resultados óptimos fue el tipo de imagen utilizada. La adición del margen extra, como en HOG, tenía una

repercusión en el sistema, en este caso negativa, el sistema, lo que dio lugar a modificarlo, y utilizar únicamente el recorte del vehículo. Las detecciones se ven afectadas notablemente, como se ha podido comprobar, y fue el punto de partida del detector Haar.

El tamaño de la imagen volvió a ser uno de los grandes parámetros a estudiar, ya que, como hemos podido observar, las bases de imágenes de los entrenamientos son bastante decisivas. En este caso, puesto que la aplicación de los filtros de Haar es notablemente invariante a la calidad de la imagen, se ha podido trabajar con el tamaño más pequeño de imagen, evitando así tener que recurrir a la ampliación de la muestra a detectar para conseguir los objetos pequeños, o lo también denominado doble chequeo, lo cual redujo el tiempo de cómputo en la detección. Cabe destacar, que a pesar de haber conseguido resultados considerablemente óptimos, este método cuenta con dos grandes inconvenientes, definiéndolos como el tiempo de entrenamiento necesario, y el número de falsos positivos que existentes.

En el caso del tiempo, estos son procesos que pueden llegar a necesitar varios días de ejercicio, por lo que la realización de las pruebas en busca de una configuración óptima puede alargarse excesivamente. Con respecto al gran número de falsos positivos que Haar introduce en las detecciones, dando lugar a un método no demasiado preciso, gracias a él, se es capaz de detectar un gran número de vehículos, pero con el problema añadido de las detecciones erróneas, lo que conlleva a utilizar un filtrado más riguroso.

En cuanto a los parámetros de detección, uno de los más relevantes vuelve a ser el ScaleFactor, el cual nos ha permitido acceder a un mayor número de detecciones, y por lo tanto, de verdaderos positivos. Esto, como se ha podido observar en la figura 4.79, eleva considerablemente el número de falsos positivos, como era de esperar de este método. Para contrarrestar este comportamiento, el filtro de las limitaciones geográficas, junto con el NMS, fueron dos herramientas clave en el análisis.

De esta forma, se puede comprobar cómo, nuevamente, se ha obtenido un método de detección de vehículos, esta vez siguiendo el ejemplo de Viola Jones [2]. Los resultados han sido bastante positivos, dentro de los límites que nuestra base de datos nos permite, debido a la resolución de la imagen, el tipo de objeto y el número de muestras. Los resultados para la configuración final son los siguientes:

Verdaderos positivos: 460
Falsos positivos: 804
Falsos negativos: 292
Posibles detecciones: 752
Precision: 36.4%
Recall: 61.2%

5 Conclusiones y trabajos futuros

El objetivo principal de este proyecto era la búsqueda de un método de detección que permitiese detectar vehículos presentes en la vía, es decir, en una situación compleja en cuanto a entorno se refiere. Para esto se ha estudiado los efectos que algunos parámetros pueden tener en los resultados finales, así como lo que ello conlleva. Con el objetivo de definir correctamente la exactitud de los algoritmos se ha recurrido al estudio de *precision* y *recall*, así como el tiempo de cómputo que este requería.

De carácter general, se ha demostrado que la combinación de HOG y SVM, obtiene mejores resultados en la detección de vehículos con cámaras infrarrojas que las cascadas de Haar, aunque sin embargo, los valores numéricos no están tan alejados. Así, para el primer método de obtiene una precision de un 52%, mientras que en el caso del Haar esta se ve afectada por la aparición de falsos positivos, propia de este método. Los Histogramas de Gradientes Orientados proporcionan un mayor nivel de detecciones, frente a las posibles existentes, cuyo valor asciende a 72.8%, mientras que para Haar este no supera 62%.

Cada método ha demostrado contar con detalles que complican los resultados óptimos, y que dificultan la llegada a un detector perfecto. Estos por ejemplo, en el caso del primer método, era la necesidad de una calidad y resolución de imagen elevada, para hacer posible la correcta extracción de los gradientes, y para Haar se corresponde al elevado número de falsos positivos que sus detecciones suponen, obligando a endurecer los requisitos, así como el procesamiento posterior de esas detecciones. Cada detector ha sobresalido en diferentes aspectos, siendo la búsqueda de elementos a poca distancia, o de gran tamaño, la fuerte potencia de HOG. En el caso de Haar, fue todo lo contrario, siendo los objetos cercanos no detectados, pero sin embargo si se fue capaz de reconocer aquellos vehículos a gran distancia.

El éxito de HOG frente a Haar es debido a la normalización de contraste que este realiza a la hora de calcular los gradientes, lo que lo hace invariable a los cambios en la imagen. Por lo general, el reconocimiento de objetos es más exacto con HOG que con Haar, ya que este último se centra en las características de textura, sin ofrecer ninguna información acerca de orientaciones, lo cual es muy relevante para la clasificación de objetos. Sin embargo, a pesar de esto, los resultados de Haar no son tan desfavorables como para no considerarlo un método valido.

De cara a futuros proyectos, se podría estudiar un método que permita una combinación de ambos procesos de extracción de característica. Además de esto, también se podría analizar la posibilidad de implementar el algoritmo con el método de aprendizaje no supervisado de Deep Learning, el cual a día de hoy está ofreciendo grandes resultados, permitiéndolo avanzar a grandes pasos.

6 Presupuesto

El presupuesto descrito incluye suministro, colocación y programación del sistema de detección en una futura incorporación en el vehículo autónomo. En él se detallan los elementos necesarios para la implementación del proyecto, los cuales corresponden al equipo informático y visual. La programación y software, así como el desarrollo completo por un ingeniero técnico, es incluido en el tercer apartado de este presupuesto. Este ha sido calculado teniendo en cuenta el número de horas necesarias para el desarrollo de este trabajo, así como el salario base de un ingeniero técnico:

$$300\ horas\ \cdot \frac{30\ euros}{horas} = 9000\ euros$$

Código	Unidad de medición	Descripción	Medición	Precio Unitario	Precio total
01		**CAPITULO 1.Equipo de trabajo**			
01.01	Ud.	**Equipo informático** Suministro y puesta a punto de ordenador personal con procesador I7-6700, 32GB de RAM DDR4. Incluido teclado y ratón óptico.	1	1200	1200
01.02	Ud.	**Cámara Visión Infrarroja FLIR Indigo Omega** Suministro y colocación de la cámara de visión infrarroja FLIR Indigo Omega Se incluye el cableado de conexión.	1	3600	3600
01.03		**Sistema de detección, software y programación** Generación y programación del sistema de detección. Incorporación en el vehículo autónomo y puesta a punto.	1	9000	9000
		Total Capitulo 1			13800

En el marco socioeconómico en el que nos encontramos a día de hoy, cabe destacar que la investigación e incorporación de este tipo de sistemas a los vehículos que circulan por nuestras carreteras es un proyecto económicamente viable, si tenemos en cuenta los avances que esto podría suponer para el usuario.

Anexos

Anexo I. Instalación de OpenCV y SVMLight en Ubuntu junto con QtCreator

El programa se ha desarrollado en el sistema operativo, Ubuntu 12.04, donde, para la instalación de los distintos programas a utilizar, así como de las librerías correspondientes, es necesaria la instalación de estos de una manera concreta.

En primer lugar se instalara QtCreator, en su versión más reciente posible, el cual se puede obtener desde la página oficial [24], y se siguen los pasos que se van indicando en cada pestaña. Tras esto, pasamos a la instalación de la librería principal con la que trabajaremos, OpenCV. Para ello que se ha seguido el tutorial [25], ya que gracias a un script que se ejecuta desde la terminal, se consigue fácilmente.

Una vez comprobada la correcta instalación, es necesaria la conexión entre el proyecto creado en QtCreator y la librería. Para ello se incluirá en el archivo ".pro" del proyecto, los siguientes comandos:

INCLUDEPATH += /usr/local/include/opencv

LIBS += -L/usr/local/lib -lopencv_calib3d -lopencv_core -lopencv_features2d -lopencv_videoio -lopencv_flann -lopencv_highgui -lopencv_imgproc -lopencv_ml -lopencv_objdetect -lopencv_photo -lopencv_imgcodecs -lopencv_stitching -lopencv_superres -lopencv_ts -lopencv_video -lopencv_videostab -lpthread -lopencv_contrib -lopencv_gpu -lopencv_legacy -lopencv_ocl -lopencv_stitching

Gracias a ellos se permite la vinculación entre el proyecto y la librería de código abierto. Se puede utilizar cualquier tipo de proyecto de los que QtCreator ofrece, siempre y cuando el lenguaje de programación, o los requisitos de este, los cumplamos.

Tras esto, se ha configurado el proyecto para poder trabajar con OpenCV. La segunda librería es la que nos permite trabajar con las SVM. Esta viene dada por un archivo ".Zip" que puede ser descargado del link ya mencionado, y solo sería necesaria su incorporación en el proyecto, así como la vinculación con este. Para este último paso, será imprescindible incluir la siguiente línea en el ".pro":

OBJECTS += svm_learn.o svm_common.o svm_hideo.o

Además, al igual que con la OpenCV, se deberá incluir el directorio de estos archivos en el apartado LIBS. Esto permitirá que el programa lo encuentre y sea capaz de hacer uso de ella, lo que en esta ocasión fue:

LIBS +=... -L/home/carmen/Escritorio/otromas/svmlight/svm_learn.o -L/home/carmen/Escritorio/otromas/svmlight/svm_common.o -L/home/carmen/Escritorio/otromas/svmlight/svm_hideo.o

Bibliografía

[1] L. d. S. Inrteligentes, «Inetligent Systems Laboratory,» [En línea]. Available: http://portal.uc3m.es/portal/page/portal/dpto_ing_sistemas_automatica/investigacion/IntelligentSystemsLab. [Último acceso: Septiembre 2016].

[2] P. Viola y M. Jones. [En línea]. Available: https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf. [Último acceso: Septiembre 2016].

[3] N. Dalal y B. Triggs. [En línea]. Available: https://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf. [Último acceso: Septiembre 2016].

[4] L. J. e. linea, «La Jornada en linea,» [En línea]. Available: http://www.jornada.unam.mx/ultimas/2015/10/19/mueren-1-25-millones-de-personas-al-ano-por-accidentes-viales-oms-4074.html. [Último acceso: Septiembre 2016].

[5] Wikipedia. [En línea]. Available: https://es.wikipedia.org/wiki/Sistemas_inteligentes_de_transporte. [Último acceso: Septiembre 2016].

[6] «CEA,» [En línea]. Available: http://www.cea-online.es/reportajes/seguridad.asp. [Último acceso: Septiembre 2016].

[7] «Honda,» [En línea]. Available: http://www.hondadreams.es/2013/03/26/tecnologia-para-la-seguridad-vial/. [Último acceso: Septiembre 2016].

[8] «Seguridad Vial,» [En línea]. Available: http://www.seguridad-vial.net/vehiculo/seguridad-pasiva/156-el-sistema-adas-ayuda-a-prevenir-accidentes-de-trafico-a-los-conductores. [Último acceso: Septiembre 2016].

[9] «nvidia,» [En línea]. Available: http://www.nvidia.es/object/advanced-driver-assistance-systems-es.html. [Último acceso: Septiembre 2016].

[10] «Toyota,» [En línea]. Available: https://www.toyota.es/world-of-toyota/safety-technology/parking-aids.json. [Último acceso: Septiembre 2016].

[11] «Toyota2,» [En línea]. Available: https://www.toyota.es/world-of-toyota/safety-technology/pre-crash-safety.json. [Último acceso: Septiembre 2016].

[12] «DGT,» [En línea]. Available: http://www.dgt.es/es/prensa/notas-de-prensa/2015/20151116-trafico-establece-marco-realizacion-pruebas-vehiculos-conduccion-automatizada-vias-abiertas-circulacion.shtml. [Último acceso: Septiembre 2016].

[13] «Wikipedia_VisionArtificial,» [En línea]. Available: https://es.wikipedia.org/wiki/Visi%C3%B3n_artificial. [Último acceso: Septiembre 2016].

[14] «coursera,» [En línea]. Available: https://es.coursera.org/learn/deteccion-objetos/. [Último acceso: Septiembre 2016].

[15] «robologs,» [En línea]. Available: http://robologs.net/wp-content/uploads/2014/05/haar-like_features.png. [Último acceso: Septiembre 2016].

[16] «bssaonline,» [En línea]. Available: http://www.bssaonline.org/content/97/5/1486/F2.large.jpg. [Último acceso:

Septiembre 2016].

[17] J. Hendriks. [En línea]. Available: https://github.com/DaHoC/trainHOG. [Último acceso: Septiembre 2016].

[18] «SVM,» [En línea]. Available: http://svmlight.joachims.org/. [Último acceso: Septiembre 2016].

[19] «saedsayad,» [En línea]. Available: http://www.saedsayad.com/images/SVR_1.png. [Último acceso: Septiembre 2016].

[20] «pyimAGENSEARCH,» [En línea]. Available: http://www.pyimagesearch.com/wp-content/uploads/2015/03/pyramid_example.png. [Último acceso: Septiembre 2016].

[21] «fewtutorials,» [En línea]. Available: http://fewtutorials.bravesites.com/files/images/fig_3_thumb.png. [Último acceso: Septiembre 2016].

[22] «pyimagesearch,» [En línea]. Available: http://www.pyimagesearch.com/2014/11/17/non-maximum-suppression-object-detection-python/. [Último acceso: Septiembre 2016].

[23] «github,» [En línea]. Available: https://github.com/Nuzhny007/Non-Maximum-Suppression. . [Último acceso: Septiembre 2016].

[24] QtCreator. [En línea]. Available: https://www.qt.io/download/. [Último acceso: Septiembre 2016].

[25] GitHub. [En línea]. Available: https://gist.github.com/willprice/c216fcbeba8d14ad1138. [Último acceso: Septiembre 2016].

[26] «slideshare,» [En línea]. Available: http://www.slideshare.net/potaters/decision-forests-and-discriminant-analysis. [Último acceso: Septiembre 2016].

[27] «El confidencial,» [En línea]. Available: http://www.elconfidencial.com/tecnologia/2016-05-20/vehiculo-autonomo-empresas-prueban-coche-futuro-google-uber_1203221/. [Último acceso: Septiembre 2016].

[28] «noticiasCoches,» [En línea]. Available: http://noticias.coches.com/wp-content/uploads/2013/11/seguridad-en-coches.jpg. [Último acceso: Septiembre 2016].

[29] «raw.github,» Septiembre 2016. [En línea]. Available: https://raw.githubusercontent.com/timlentse/Add-Salt_Pepper_noise/master/add%20noise%20%20image.png.

[30] «images.slideplayer,» Septiembre 2016. [En línea]. Available: http://images.slideplayer.es/12/3421896/slides/slide_10.jpg.

[31] A. d. l. E. Hueso, «Perception Systems _Class 2: Elements,» 2016.

[32] «pce-iberica,» [En línea]. Available: http://www.pce-iberica.es/medidor-detalles-tecnicos/images/camara-termica-pce-tc6-uso-10.jpg. [Último acceso: Septiembre 2016].

[33] «saunasInfrarrojos,» [En línea]. Available: http://www.saunasinfrarrojo.com.ar/archivos/far_Infrared.jpg. [Último acceso: Septiembre 2016].

[34] «latermografia,» [En línea]. Available: http://www.latermografia.com/2011/las-

paletas-de-la-camara-infrarroja. [Último acceso: Septiembre 2016].

[35] «scielo,» [En línea]. Available: http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0120-56092007000300013. [Último acceso: Septiembre 2016].

[36] «teleformacion,» [En línea]. Available: http://teleformacion.edu.aytolacoruna.es/FISICA/document/fisicaInteractiva/OptGeometrica/Instrumentos/Microscopio/puntoProx.gif. [Último acceso: Septiembre 2016].

[37] F. Lecumberry, «researchgate,» [En línea]. Available: https://www.researchgate.net/publication/228728583_Calculo_de_disparidad_en_imagenes_estereo_una_comparacion. [Último acceso: Septiembre 2016].

[38] «gsyc.urjc,» [En línea]. Available: https://gsyc.urjc.es/jmplaza/carcounter.png. [Último acceso: Septiembre 2016].

[39] Blog.Infaimon. [En línea]. Available: http://blog.infaimon.com/wp-content/uploads/2011/12/llantes2.jpg. [Último acceso: Septiembre 2016].

[40] «Pybonacci,» Septiembre 2016. [En línea]. Available: http://pybonacci.github.io/scipy-lecture-notes-ES/_images/plot_lena_denoise_1.png.

[41] «img.ar.class,» [En línea]. Available: http://img.ar.class.posot.com/es_ar/2015/05/04/Monocular-Vision-Nocturna-Marca-Russia-En-Excelente-Estado-20150504041339.jpg. [Último acceso: Septiembre 2016].

[42] N. M. Cruz, «Desarrolo de un sistema avanzado a la conduccion en tiempo real para la deteccion de peatones en entornos urbanos complejos,» Madrid, 2013.

[43] J. M. B. Cava, «Deteccion de peatones utilzando camaras de infrarrojos,» Madrid, 2014.

[44] R. M. Manso, «Sistema de visión artificial para la detección y lectura de matrículas,» Valladolid, 2014.